AFRL-IF-RS-TR-2004-122
Final Technical Report
May 2004

SENSOR AGENT PROCESSING SOFTWARE (SAPS)

BAE Systems Integrated Defense Solutions

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. H579

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-122 has been reviewed and is approved for publication

APPROVED: /s/
RICHARD C. BUTLER II
Project Engineer

FOR THE DIRECTOR: /s/
WARREN H. DEBANY, JR.
Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	MAY 2004	FINAL Jun 99 – Aug 03

4. TITLE AND SUBTITLE

SENSOR AGENT PROCESSING SOFTWARE (SAPS)

5. FUNDING NUMBERS

C - F30602-99-C-0159
PE - 62301E
PR - H579
TA - 16
WU - 01

6. AUTHOR(S)

Steve Beck
Joe Reynolds

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

BAE Systems Integrated Defense Solutions
6500 Tracor Lane
Austin TX 78725

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

AFRL/IFGA
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRLIF-RS-TR-2004-122

11. SUPPLEMENTARY NOTES

DARPA Program Manager: Sri Kumar/IPTO/(703) 696-0174
AFRL Project Engineer: Richard C. Butler II/IFGA/(315) 330-1888 Richard.Butler@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

The Defense Advanced Research Projects Agency (DARPA) Information Technology Office (ITO) sponsored the Sensor Information Technology (SensIT) program to develop and then demonstrate technologies for large wireless networks. Numerous commercial firms, universities, and research laboratories were given contracts to develop specific technologies required to support the overall objectives. Products from each organization were combined in a common architecture to demonstrate the capability of large wireless networks when applied to a large number of networked sensors collaborating to provide battlefield intelligence. BAE Systems, under contract with Air Force Research Laboratory (AFRL), was primarily responsible for the Sensor Agent Processing Software (SAPS).

This report describes the scientific and technical work performed by BAE Systems under Contract F30602-99-C-0159. This includes design and development of the processor application software architecture, development of signal processing algorithms, and implementation of SAPS architecture to support numerous field tests and demonstrations. The report also describes how the BAE Systems' SAPS was combined with other key technologies developed by other SensIT participants to demonstrate that the technologies exist to effectively deploy and utilize large scale sensor networks. The report also addresses efforts undertaken by BAE systems to leverage the SensIT R&D success to develop sensor networks that can be deployed and used in environments such as Afghanistan and Iraq.

14. SUBJECT TERMS

Distributed Sensor Networks, Wireless Networks, Collaborative Signal Processing, Target Detection, Tracking, Low Power Processing, Low Power Communications, SensIT Nodes

15. NUMBER OF PAGES 79

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

The authors sincerely appreciate the efforts of Carl Bott and Charles Kiers to provide insight and understanding into the vital role of sensor networks in battlefield environments.

1. INTRODUCTION and OVERVIEW

The purpose of the DARPA SensIT program was to develop and then demonstrate technologies for large wireless networks. The application chosen to highlight the networking capabilities was sensor systems; fields with large numbers of sensor nodes collaborating to provide intelligence. A number of leading experts from industry, government laboratories, and academia were selected to perform research and develop applications for the nascent field of wireless sensor networks. The eventual goal of the program was to demonstrate technology that could be massively deployed, self configured, and run autonomously, sensing and making decisions for a long period of time.

The original contractors that started on SensIT included:

BBN Technologies	- Integrator
Sensoria	- Hardware for wireless sensor nodes
BAE Systems	- Sensor agent processing (detection and tracking) software
USC-ISI	- Diffusion routing networking software
MIT-LL	- Dynamic declarative networking software
UCLA	- GRAB self configuring sensor net protocols, web caching
Cornell University	- Cougar object relational database and query
University of Maryland	- Declarative sensor tasking language
Penn State University Applied Research Laboratory (PSU-ARL)	- Mobile code and collaborative tracking
NAI	- Micro-encryption and network security

BAE Systems, through Sensor Agent Processing Software (SAPS), was chosen to lead the design of the processor application software architecture and to develop sensor signal processing algorithms to perform signal enhancement, target detection, classification, and tracking. Because of the distributed nature of the problem, one of the primary research issues was how to make best use of the processing capabilities available on a network of nodes under the constraints of low power processing and very low power and low bit rate communications. The term ‘collaborative processing’ was used to describe algorithm research under those constraints.

BAE Systems developed algorithms for three sensing modes; acoustic, seismic, and passive infrared (PIR) motion sensing. The algorithms reported time of Closest Point of Approach (CPA). Additionally, BAE Systems designed and developed algorithms for node resident data persistence. These were in the form of Data Repositories. The repositories became a key capability in defining collaborative processing algorithms, as each stage of signal processing on each node and shared across the network was retained for associated utilization. BAE Systems implemented these algorithms on two versions of the SensIT nodes supplied by Sensoria Inc., initially under Microsoft Windows CE and later under Linux. Additionally, BAE Systems participated in four field exercises, supplying processing software in support of all other teams. BAE Systems also conducted data collection activities with proprietary equipment and an extended sensor suite including µradars, magnetometers, and improved acoustic capability.

2. BAE Systems SAPS PROGRAM HIGHLIGHTS

Date	SAPS Program Highlights
July 1999	SensIT kickoff at DARPA in Washington DC
Sept	BAE collects and processes outdoor voices for detection
Oct	SensIT PI Meeting in Marina Del Rey CA.
Nov	BAE participated in CAX at 29 Palms, collect acoustic vehicle signatures
Dec	Receive WINS 1.0
Jan 2000	SensIT Integration meeting in Boston MA
Feb	Devise repository architecture, begin implementation
Mar	SensIT PI Meeting in Boston MA.
Apr	Begin detection algorithm implementation
May	Begin SITEX00 planning
June	Begin software integration on WINS 1.0
July	SITEX00 at 29 Palms CA. First demonstration of collaborative processing.
Sept	Begin SITEX00 data analysis
Oct	SensIT PI Meeting in Honolulu, HA
Nov	Begin multi-modal sensor detection analysis
Dec	Study low power processor architectures and processing capabilities
Jan 2001	Begin imager integration and system testing for SITEX01.
Mar	SITEX01 at 29 Palms CA. Demonstrate collaborative target tracking on four nodes using multi-modal detection and an imager.
Apr	SensIT PI Meeting in St. Petersburg, FL
May	Begin code conversion to Linux
July	Publish BAE Low Level Detection Processing scheme
Aug	Publish BAE Detection API
Sept	Test code on WINS 2.0
Oct	Participate in code integration in Boston
Nov	SITEX02 at 29 Palms CA. Demonstrate collaborative detection and tracking on a large field of nodes.
Jan 2002	SensIT PI meeting in Santa Fe NM. Demonstrate collaborative tracking in a room and outdoors in a parking lot.
Feb	Begin human presence detection algorithm study
Mar	Begin BAE low power wireless node design
May	Purchase Ember nodes and begin integration
July	Demonstrate in the building human detection and tracking
Aug	Demonstration with UTK, Auburn, and Penn State BAE Testbed
Sept	Begin classifier algorithm for vehicle of interest
Oct	Collect data for vehicle classifier study
Nov	SensIT PI Meeting in Boston.
Dec	Set up additional nodes in the building at BAE Testbed

3. SensIT DATA ARCHITECTURE

Data architecture, specifically data persistence and sharing, was a design focus considered central to the network collaborative processing architecture (Figure 3-1). Several items went into choosing the final repository design:
[1] the experimental nature of the program called for retaining and sharing as much of the raw and intermediate forms of data as feasible,
[2] radio data rates and time synchronization capability indicated that sharing raw time series was likely infeasible, but this was reexamined with the second hardware version,
[3] sharing retained data should be closely tied to the capabilities of the network Diffusion data exchange algorithm.

Sensor systems have several scheduling and processing regimes, usually structured in software as separate tasks or processes with their own resource allocation and scheduling priority settings. One task is tightly coupled to sensor data input timing and constrained by sensor physics phenomenon. Another task is signal and information processing, which must run in real-time, but are not tied to sensor hardware. Finally, there is a task set which is determined by tactical activity.

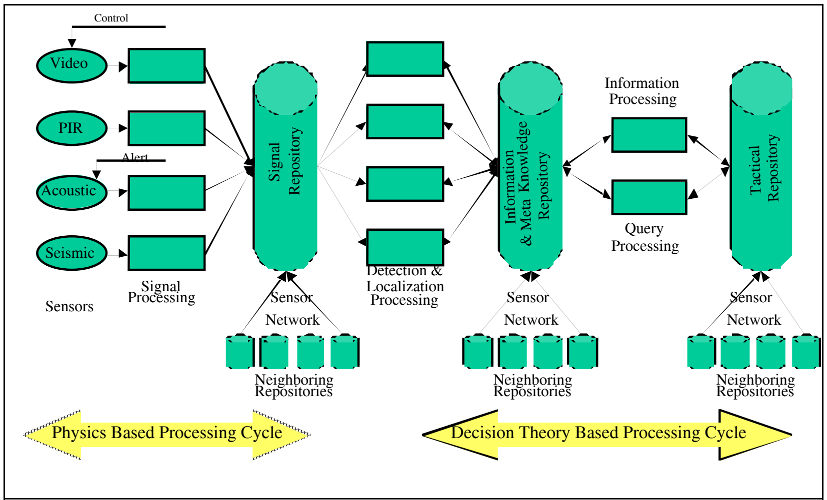


Figure 3-1 SAPS Processing and Data Persistence Architecture

A data repository class was implemented and used to retain progressive stages of processing. A Time Series Repository stored minimally processed sensor samples and facilitated sharing data among SensIT teams that also developed signal processing capability. The Time Series Repository isolated other teams from details of node hardware and design changes that occurred. It also reduced software congestion by having a single thread interface with the sensor interface.

Other repository instances followed the successive stages of sensor processing, retaining more and more abstract data items. After the time series was low level processed data, such as power spectra or normalized filter time series. Then, there were detection events, and finally, tracks or composite target identifications. The repositories proved a valuable technique to store and share data. Non-signal repositories were developed; a node status repository to retain time dependent descriptors of the node, and a meta-knowledge repository which retained items such a target characteristics, query, and tasking settings.

Repository Architecture & Functionality

Repositories are a mechanism for sharing data on a node among the various tasks and de-coupling their interaction and timing dependencies. Repositories are a mechanism for sharing data among nodes of a sensor field. Repositories buffer the data to enable "look-back" processing and node-node collaborative processing. (Since target kinematics usually means that what's "now" at one node is "then" at another node, the time synchronization facility of the repositories is extremely important in collaboration.)

The BAE Systems repositories are designed to provide an efficient means of storing and retrieving real time data that is shared between several independent computing processes. Repositories are defined in tasks that supply data and fill the repositories, and tasks which use the data subscribe to the repositories to gain access. The tasks utilize operating system specific signalizing tools to share a common repository. The repositories were interfaced with both the MIT-LL and USC-ISI versions of DIFFUSION Data Routing. In order to interface with the repositories, a small static link library was written by BAE Systems. This library provides the following functionality:

1. Allows a user to connect to a repository, or automatically create a newly initialized one, if there are currently no users.
2. Add data to a repository.
3. Get the repository header.
4. Retrieve data from a repository.
5. Get a copy of the newest record.
6. Get the position of the newest record.
7. Retrieve historical data.

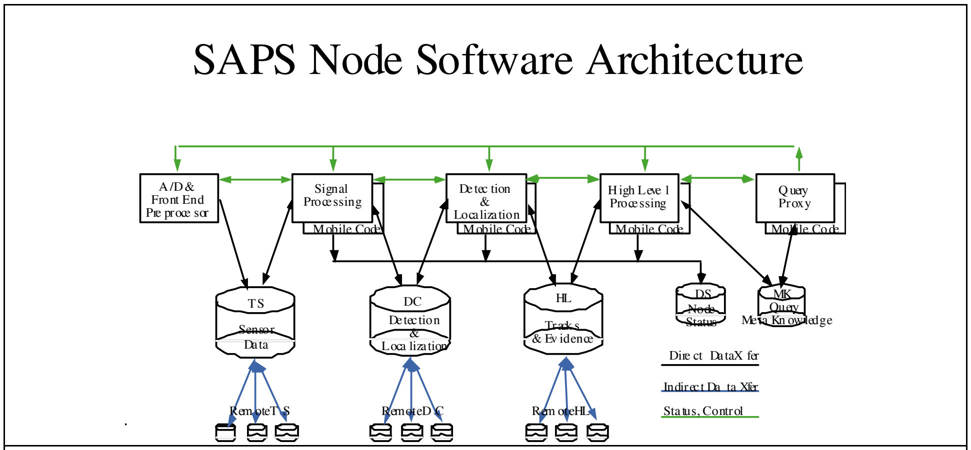


Figure 3-2 SAPS Node Network Processing Architecture

The conceptual architecture was implemented on both SensIT sensor node versions (Figure 3-2). WINS 1.0 nodes had serious hardware limitations due to the I/O between the processor and the sensors and radios. This was compounded by the scheduling limitations of the WinCE operating system. The implementation on WINS 1 nodes is shown in Figure 3-3.

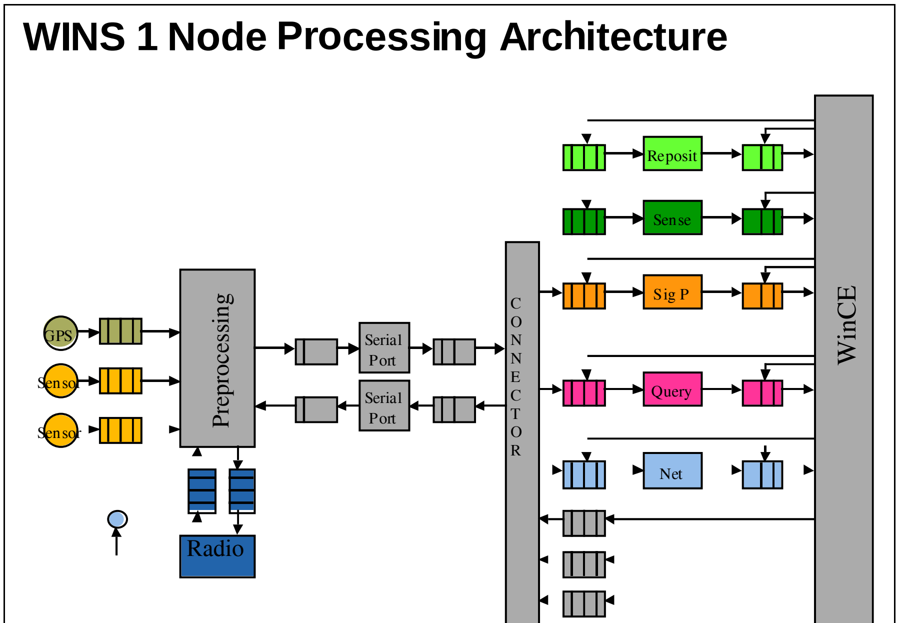


Figure 3-3 WINS 1.0 Node Hardware Processing Architecture

These limitations were addressed during the first year of SensIT and corrected in the design of WINS 2.0 nodes. Effective network communication rates increased by a factor of 500 between versions. Scheduling restrictions ceased to be a problem. The architecture implementation shown in Figure 3-4 approached the ideal.

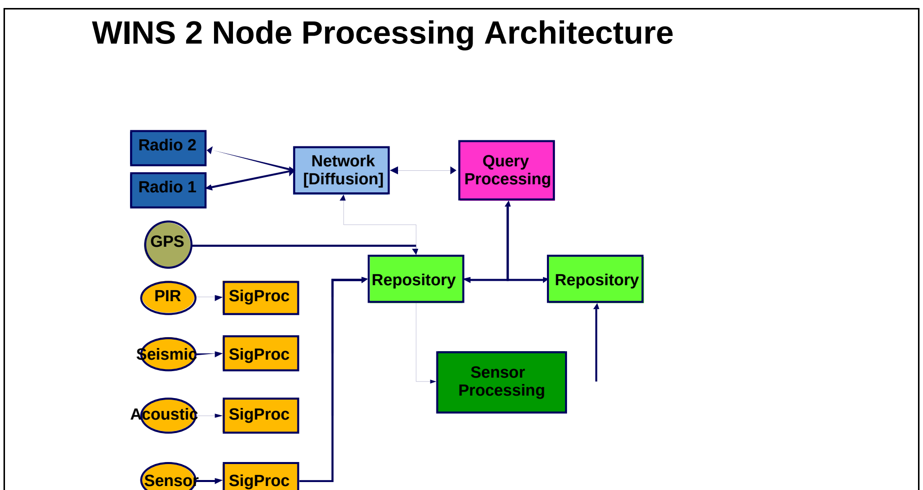


Figure 3-4 WINS 2.0 Node Hardware Processing Architecture

4. LOW LEVEL PROCESSING FOR DETECTION

BAE Systems developed and provided signal processing software matched to the common research goals to all other SensIT participants. The processing was delivered on WINS 1.0 nodes in time for the SITEX00 experiments at MCAGCC, Twentynine Palms, CA, in July/August 2000. It was used by MIT-LL, with its version of DIFFUSION, and jointly by USC-ISI, with its DIFFUSION, by BAE Systems, and by PSU-ARL in a Collaborative Detection experiment. Revised software was again delivered in September 2001 on WINS 2.0 platform for integration and use by all participants in SITEX02, held at Twentynine Palms during October/November 2001.

Process flows for the algorithms are shown below in Figures 4-1 through 4-3. Details of filters referenced are described by Figure 4-4.

Low Level Acoustic Processing

Pre-Processor Channel	1/4	
Pre-Processor Sampling Rate	4960.32 Hz	
Pre-Processor Buffer Size	256 Samples	
FFT Size	1024	or 992+32 zeros
FFT/Detection Rate	4.8440625 Hz	or 5.000322581 Hz

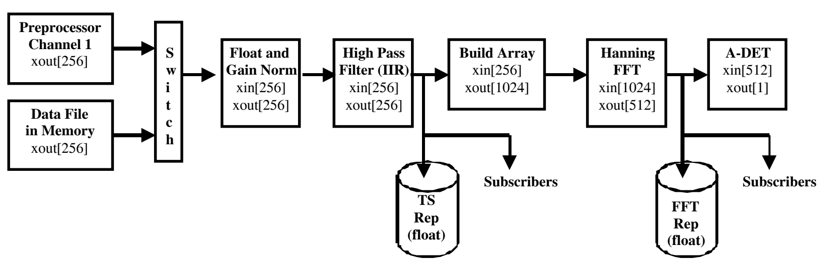


Figure 4-1 Low Level Acoustic Processing Flow

Low Level Seismic Processing

Pre-Processor Channel	2/4	
Pre-Processor Sampling Rate	4960.32 Hz	
Pre-Processor Buffer Size	256 Samples	
Downsampled Rate	496.032 Hz	
FFT Size	256	or 248 + 8 zeros
FFT/Detection Rate	1.937625 Hz	or 2.000129032 Hz

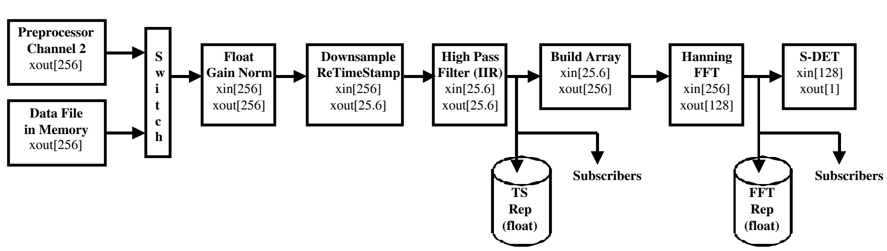


Figure 4-2 Low Level Seismic Processing Flow

Low Level PIR Processing

Pre-Processor Channel	3
Pre-Processor Sampling Rate	4960.32 Hz
Pre-Processor Buffer Size	256 Samples
Downsampled Rate	49.6032 Hz
Detection Rate	49.6032 Hz

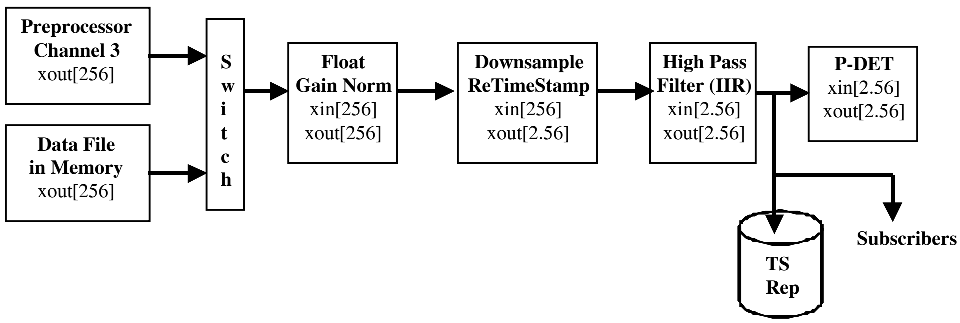
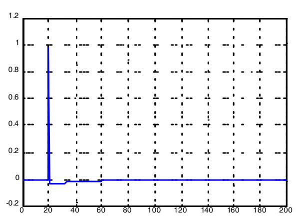


Figure 4-3 Low Level PIR Processing Flow

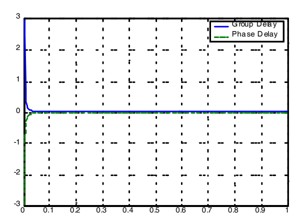
IIR High Pass Filter

Butterworth Order 1

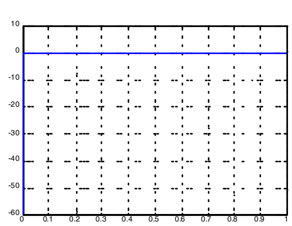
Time Domain
Impulse Response
at sample number 20



Phase and Group
Delay as a function
of normalized freq.



Frequency Domain
Impulse Response
- Normalized Freq.
(Nyquist = 1)



Frequency Domain
Impulse Response
- Zoom on Frequency

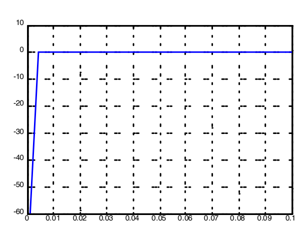


Figure 4-4 Filter Descriptions

Slower channels were down-sampled due to restrictions in the WINS 2.0 node A/D software interface, which imposed performance penalties for different sample rates on different channels. So, the rates were reduced by filtering and retimestamping before insertion into the Time Series repository. This is shown in Figure 4-5 & 4-6.

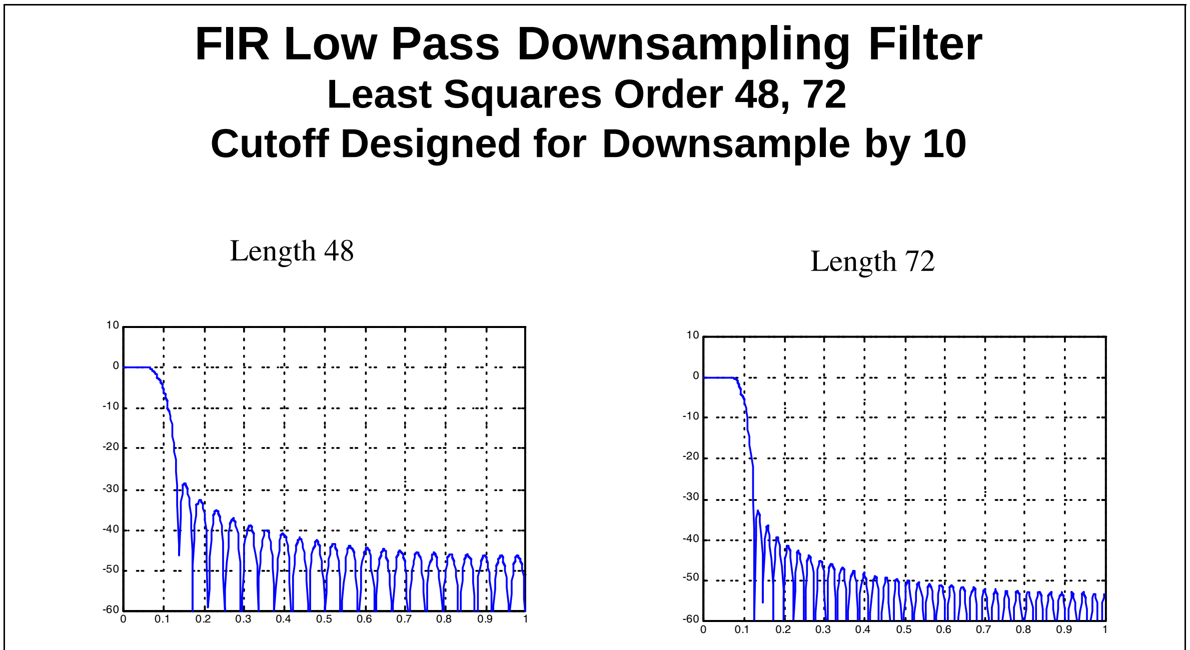


Figure 4-5 Down Sample Filters

ReTimeStamp Processing

- The downsampling process is accomplished by low pass filtering the data, and then selecting every DS sample.
- The filtering is accomplished with an FIR filter of length NF.
- The time stamp will account for the group delay of the FIR filter.

E.g. NF=8, DS=4

x = 0123456789ABCDEF0123...

TS=3

TS=7

TS=11

For a single downsampling filter of Length=NF, and downsampling rate = DS

$$TS(i) = NF/2 - 1 + i * DS \quad \text{for downsampled output index } i = 0,1,2...$$

For a two stage downsampling process with the following parameters:
Stage 1 = NF1, DS1 and Stage 2 = NF2, DS2

$$TS(i) = (NF1/2 - 1) + (NF2 * DS1/2 - DS1) + (i * DS1 * DS2)$$

$$\text{for downsampled output index } i = 0,1,2...$$

Figure 4-6 Time Stamp Processing of Down Sampled Data

5. EVENT PROCESSING

SensIT common experiment and demonstration goals involved detection (and tracking) of various vehicle targets and query processing to report the detections. Detection and reporting time of Closest Point of Approach (CPA) was consistent with the goals. High Pd and low Pfa algorithms for this were developed and delivered. Other groups used the low level signal processing capabilities and developed bearing estimation algorithms, both single-node and multi-node collaborative. The CPA detections were used by BAE Systems in implementing a distributed Kalman Tracking capability demonstrated at SITEX01. Xerox PARC developed a probabilistic tracker using signal strength estimates and showed it at the final PI meeting. PSU-ARL and Fantastic Data each developed a tracker using low level signal repository data. Several groups attempted classifiers.

BAE Systems based the detection processing on signal physics, on kinematic constraints, and on analysis of field data from WINS sensors. Sensors were characterized in BAE Systems laboratories and calibration settings were used in processing and shared with other groups as requested. Significant differences were found with some microphones and with most PIR sensors; these were adjusted by Sensoria Inc.

Event processing for Acoustic, Seismic, and PIR are shown in Figures 5-1 through 5-3.

Acoustic Detection Processing

The acoustic detection processing is based on the scheme proposed at the April 2000 PI meeting in Boston.
Several changes were necessary due to changes in the sampling rate and to account for heuristics.
The outputs of detection processing are stored in the High Level Event Detection Repository.

Input data rate	= 5000 Hz
Detection frame rate	= 5 Hz
Detection time resolution	= .2 seconds
Detection latency	= 2 seconds

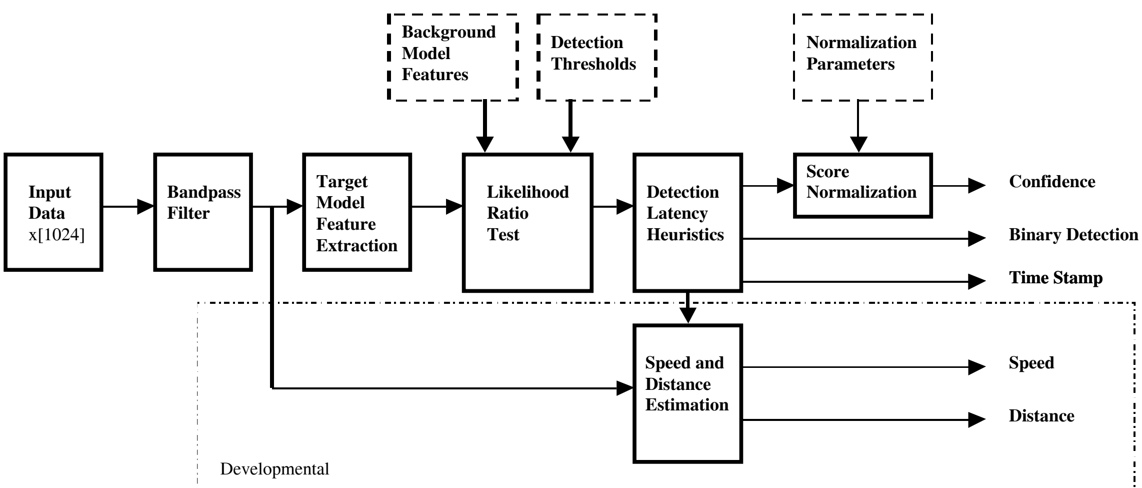


Figure 5-1 Acoustic Detection Processing Flow

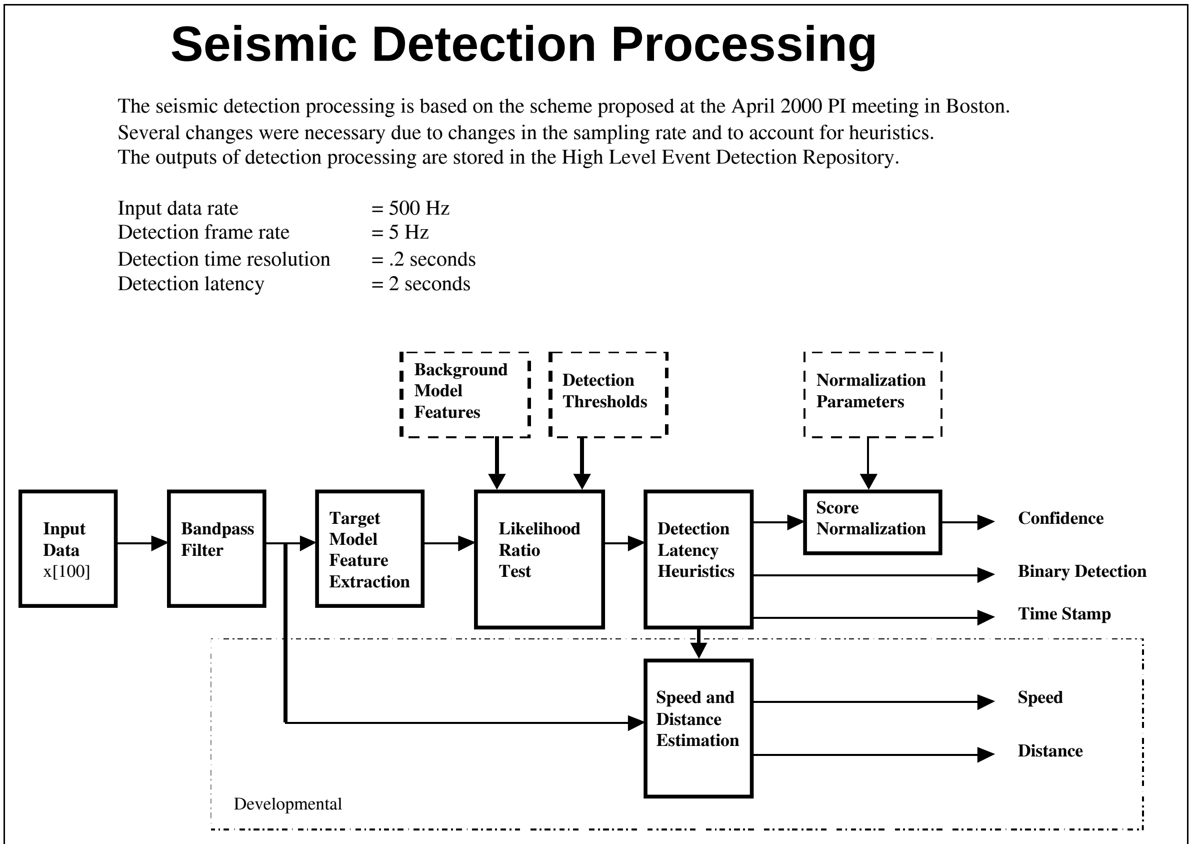


Figure 5-2 Seismic Detection Processing Flow

PIR sensors are attractive low power, low cost sensors. Their data rate is low, enabling them to be processed by very simple processors. In support of human presence detection experiments, BAE Systems studied PIR sensors as initial cueing devices. PIRs had been used under a variety of outdoor conditions: summer desert, spring desert, spring wet/rain, and winter freezing. Indoor controlled tests were run to characterize performance. The experiment conditions and performance curves are shown in Figure 5-4 and Figure 5-5. The WINS sensor units were temperature characterized to support a wider range of operation; false alarms were more numerous below 20F. Also note that the PIR mechanism lends itself to determining the direction of target motion.

PIR Detection Processing

The PIR detection processing is based on the scheme proposed at the April 2000 PI meeting in Boston.
Several changes were necessary to account for heuristics.
The outputs of detection processing are stored in the High Level Event Detection Repository.

Input data rate	= 50 Hz
Detection frame rate	= 5 Hz
Detection time resolution	= .2 seconds
Detection latency	= 2 seconds

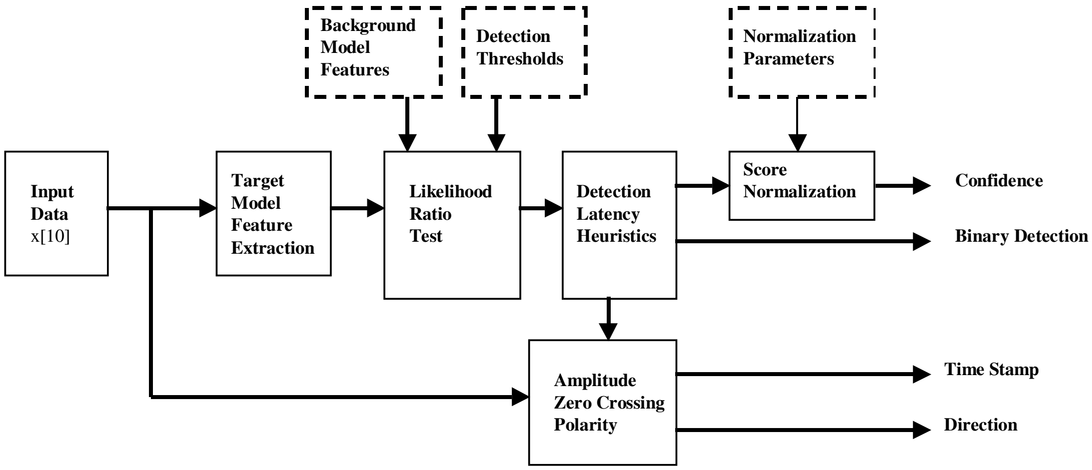
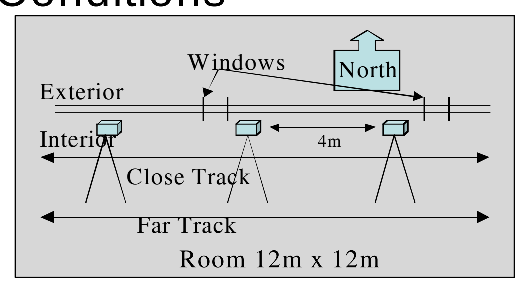


Figure 5-3 PIR Detection Processing Flow

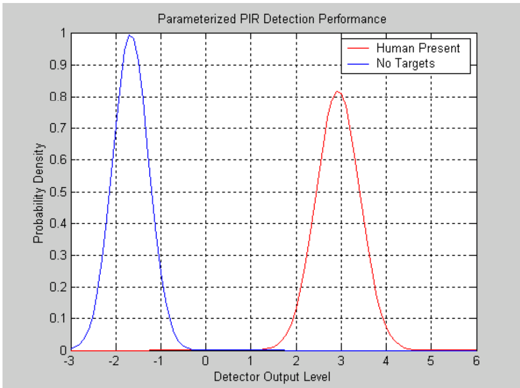
Study Conditions

- Three nodes deployed on the floor, along wall in a vacant lab space with 4m spacing
- Sensors arranged so beams are horizontal = 0¼, 25¼, and 45¼
- Working area blocked from sensors by 5'6" panels, cabinet blocks view of doorway
- Lights are Off, room has windows
- Targets are four different people, from 6'2" to 5'4"
- Tracks are at two distances to the sensors to correspond to typical rooms and hallways
  - 1.5m
  - 4m
- Tracks are at three speeds
  - a slow pace like a sneak search
  - A normal walk
  - Slow run/jog

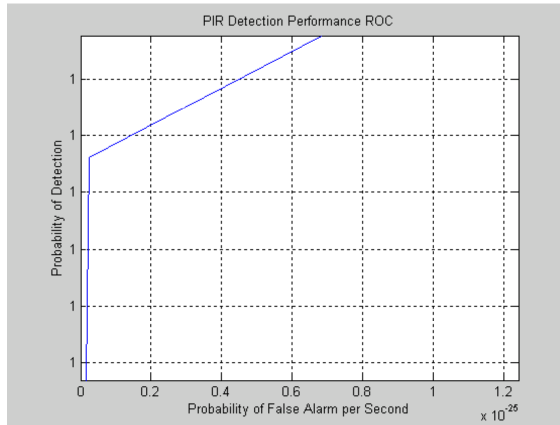


- **PIR Observed Performance**
  - **Spring Desert Conditions**
    - **Vehicles - 20/60m**
    - **People - 20m**
  - **Summer Desert Conditions**
    - **Vehicles - 60m**
    - **Troops - 30m**
  - **Fall Desert Conditions**
    - **Vehicles - 120m**
    - **Troops - 30m**
  - **Winter Urban Conditions**
    - **Vehicles - 20m**
    - **People - 15m**
  - **Indoor**
    - **Space Limited**

Figure 5-4 PIR Performance Characterization Study



PIR ROC for Indoor Detection



Pdet = 1

Pfa = 1x10**-25

Figure 5-5 PIR Performance

6. COLLABORATIVE TRACKING

6.1 Vehicle Tracking

BAE Systems developed a distributed collaborative vehicle tracker for a demonstration experiment at MCAGCC, 29 Palms, in March of 2001 at SITEX01. The tracker was to cue an imager and the experiment and node lay-down is discussed in Section 9.

The tracker was a two dimensional Kalman tracker which used time of target closest point of approach (CPA) at nodes in the sensor field to estimate Latitude and Longitude components of motion. Nodes collaboratively built a map of the sensor field at power-on. Each node measured its location by GPS using UTM coordinates, which are a localized Cartesian plane. (Due to the limited extent of the sensor field, altitude was ignored.) When a node had successfully localized itself, the Diffusion data exchange facility was used to share its status and configuration, including sensor compliment, imager capability, and UTM coordinates, with other nodes in the area of the sensor field. Each node subscribed to CPA events from neighboring nodes. Thus, as a target entered the field, nodes at the entrance point could determine it was a new target due to their edge position and lack of a CPA corresponding to the target. The first node (or nodes in the unlikely case of simultaneous detection) created a track and published to Diffusion their Node-ID, Time of CPA, and Track ID. Subsequent detecting nodes associated their CPA with an existing track based on vehicle kinematic characteristics, and published their track packet. Simple target speed was usually sufficient to correctly link a CPA with active tracks, but track association could be augmented with meta-knowledge of roads through the field or other a-priori weights for a Bayesian decision algorithm. Each node in the field subscribed through Diffusion to receive track packets. So, as CPA events were published, each node could independently cycle a local copy of the Kalman Tracker on receipt of the track packet from Diffusion. Tracks were deleted by nodes at the edge of the sensor field or when they failed to be updated for a period of time. The track packets were 54 bits in length, consisting of the minimum information: Packet-Type, source Node-ID, and Time. The imager node used the track estimate of time-in-field-of-view to control its digital camera, which had a 5 second latency from trigger to image capture.

During operation a track table maintains a sequence of track points, described below, and experiment specific binary data defined inside associated algorithms, but allocated in the track table.

Track point layout:
{ common information that can be provided to and handled by a collaborating user. }
id string, { id of track, arbitrary text }
time double, { time of detection, UTC, seconds since 1jan70 (system clock) }
{ target state information }
zone byte, { UTM zone reference, -1 means unknown }
pnorth double, { estimated location, meters north of reference point }
peast double, { estimated location, meters east of reference point }
pvalid boolean, { T=location was estimated; F= location not estimated}
vnorth double, { estimated velocity in m/s, northerly component}
veast double, { estimated velocity in m/s, easterly component }

vvalid boolean, { T=velocity was calculated; F=velocity not calculated}
class byte, { target classification}
cvalid boolean, { T=classification computed; F=classification not computed}
{ tracker-specific information follows}
{ GUI will not understand without augmentation }
typecode string, {specifies tracker-type}
trackspec blob {tracker-specific fields – for example, this is where confidence }
{ or accuracy values would go, covariance matrices, probability }
{ grid, etc.}

Notes:
This layout is called **Track point** because typically, a sequence of such records, all with the same ID, makes up a target path (track). Thus, **id** is an id for a track, and each track point will carry that id for accumulating the track itself. All trackers must provide some sort of id for a track. The assumption will be that multiple track points with the same id are points on the same track.
Defaults and assumptions by the GUI are:
time is the time a vehicle is at the estimated location. Default 0 (won't be displayed)
Zone, pnorth, peast will be valid UTM values (in the zone the GUI is working) whenever pvalid is True.
Vnorth,veast are estimates of speed of the target identified by zone, pnorth, peast, time. Default 0,0.
Classification codes are defined below.
pvalid, vvalid, cvalid have default values of False.
example of trackspec for typecode = "Kalman"
bloblength double,
cov11 f32, { covariance matrix from Kalman filter }
cov12 f32,
cov13 f32,
cov14 f32,
cov22 f32,
cov23 f32,
cov24 f32,
cov33 f32,
cov34 f32,
cov44 f32
Classification codes
These codes were used in a joint experiment with Penn State University Applied Research Laboratory (PSU-ARL); note that codes 0-9 are features and 10 and up are target types. PSU-ARL provides features (weight and method of locomotion) in addition to a code book value (this additional information would go in their blob) – although any of these code values could be used by a track display. For example, if we just know that the target is wheeled, the GUI could display just that information.
UNKNOWN=-1;
WHEEL=0;

TRACK=1;
LIGHT=2;
HEAVY=3;
Buzzer=10;
Motorcycle=11;
TruckGas=12;
TruckDiesel=13;
BuzzerRed = 14;
BuzzerBlue = 15;

6.2 Personnel Tracking

As part of a focus on sensing support for small fire teams in complex terrain, BAE Systems adapted the vehicle tracking algorithm to use inside buildings. The Kalman algorithm was kept, but default kinematic assumptions were set for humans walking, instead of the SITEX01 vehicle kinematics. Constraints were applied to the association of CPA with Track-ID, based on a hallway’s allowed paths and any node sequencing which hallways provide. The person tracking experiments investigated the feasibility of determining a target’s Red/Blue classification by maintaining movement history and feasibility of inferring control of an area by relative population count. The experiments are discussed in Section 12. Associated efforts to detect and identify humans are presented in Section 7.

7. HUMAN PRESENCE DETECTION

BAE Systems conducted a series of experiments in detecting human presence and distinguishing people from animals and natural phenomena, and to detect human presence when masked by vehicle noises. Some of the efforts were conducted under other projects, such as APLA Track 4 (the RATTLER Project), but jointly used in SensIT or with SensIT data sets, or sharing portions of SensIT developed algorithms, or verified during SensIT experiments. These efforts eventually led to SensIT partially funding the development of a special miniature, low power sensor node specifically for human detection. The node is described in Section 8.

Sensing Modes:

Microradar – a very low power ultra wideband microradar was used during RATTLER (May 2001) to detect and identify distinctive human activities, such as walking, sweeping a mine detector, and crawling. The data used was limited and from local sources. The radar was again used during SITEX02 to collect signature data on all target vehicles and on two battalions of marching and lounging Marines.

Magnetometer – a miniature magnetometer was developed by BAE Systems for RATTLER, based on a design from University of California, Berkley, which was used by Kris Pister of Berkley under SensIT during SITEX01 (March 2001). While the magnetometer suffers from transducer issues, has limited detection range, and has a high power draw, it is a perfect indicator of human presence. Only humans, or domesticated animals associated with humans, move metal and are thus detectable by a stationary magnetometer. The sensor developed uses a small (18 pin flat pack) 2-axis magneto resistive magnetometer from Honeywell. The analog interface nulls the Earth's field (thus providing a compass) and amplifies the sensor output with a 0.1Hz-2.0Hz bandpass. This response is sufficient to resolve human movement over detectable ranges.

Passive Infra Red Motion Detector – The PIR was a primary SensIT sensing mode and proved useful in human presence detection. BAE Systems was able to develop a highly reliable (Pd approaching 1 and Pfa of 10^{-25}. The PIR is a cueing sensor for other classification modes. The PIR performance is described in Section 5.

Acoustic – Human speech is a good indicator of people. An algorithm was developed under IRAD and ported to the SensIT Sensoria WINS 2.0 nodes and to the Wireless Sensor Node described in Section 8. The original goal for the node was to detect humans in quiet environments, with speech a prime indicator.

Seismic – Seismic activity is a useful discriminator of weight and use of a seismic sensor in conjunction with a simple PIR cueing sensor allows a sensor to ignore small animals. Walking, by un-alerted people or animals, produces distinctive patterns. These must be distinguished from rain drops and by themselves have a moderately high Pfa, but they provide a powerful confirming indicator in combination with other sensing.

8. BAE Systems WIRELESS SENSOR NODE

Following the SensIT PI meeting in Santa Fe, NM, BAE Systems was tasked to build a limited number of sensor nodes intended to detect human presence in sparse desert conditions. The architecture of the nodes was unspecified, but it was expected to be built up on the SensIT experience. BAE Systems constructed eight nodes during the February to July interval. Integration was delayed waiting on early deliverable of Ember Corp radios. The node hardware overview and architecture are shown in Figure 8-1. The algorithms for motion and seismic activity were based on SensIT work. Speech detection, vehicle identification, footstep detection, and magnetic processing were based on BAE Systems algorithms. The radio supplier, Ember Corp, was a SensIT participant with new venture capital.

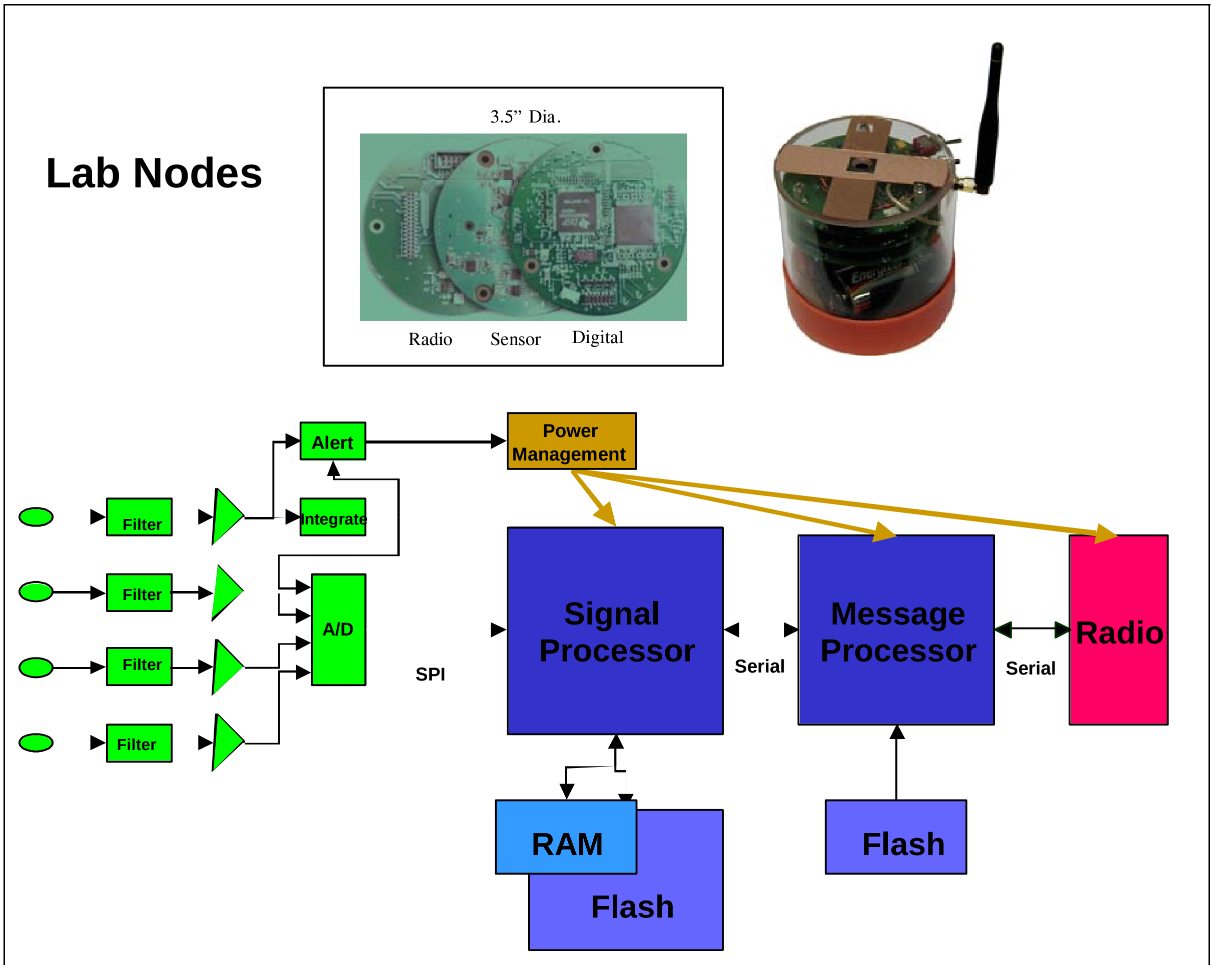


Figure 8-1 BAE Systems Wireless Sensor Node Overview

Several sensors are accommodated on the node, with acoustic, seismic, and a motion detection mode chosen for initial implementation. The nodes have a power cycling 'sleep' mode with an analog wake-up signal. A long period filter measures background, and if short term signals exceed a LLR threshold, the unit is activated. It can begin processing immediately by using the background filter as its noise estimate.

The units were designed to maximize battery life and can run indefinitely on 4”sq of solar cells and rechargeable batteries. They use a low power signal processor chip. It digitizes sensor time series with an A/D via an SPI interface and DMA transfer to RAM. Processing of the various channels and multi-node collaboration is shown in Figure 8.2 through 8.5. Alerts are passed via an SPI interface to an Ember radio, which links individual sensors to a gateway node for a final link to users. The power design is successful and battery life when fully active, not duty cycling, exceeds 5 days on D-cell batteries. Approximately 30,000,000 contact identifications are accommodated, with 50,000 radio reports.

The radios automatically form a mesh network. Redundant routing and load sharing are implemented. The mesh router allows operation in complex terrain, which blocks single link networks. The radios are direct sequence spread spectrum in the 915MHz band and follow 802.15 standards. Reports and radio transmissions are independently encrypted with AES 128bit algorithms.

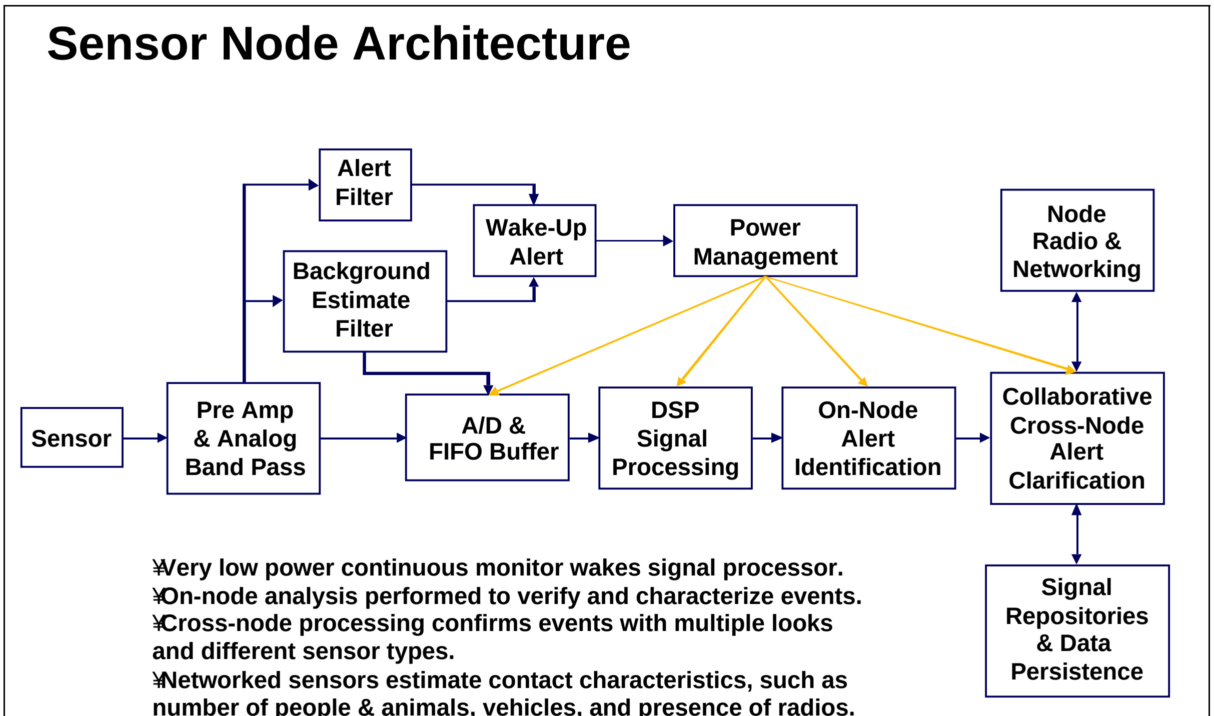


Figure 8-2 BAE Systems Wireless Sensor Node Functional Architecture

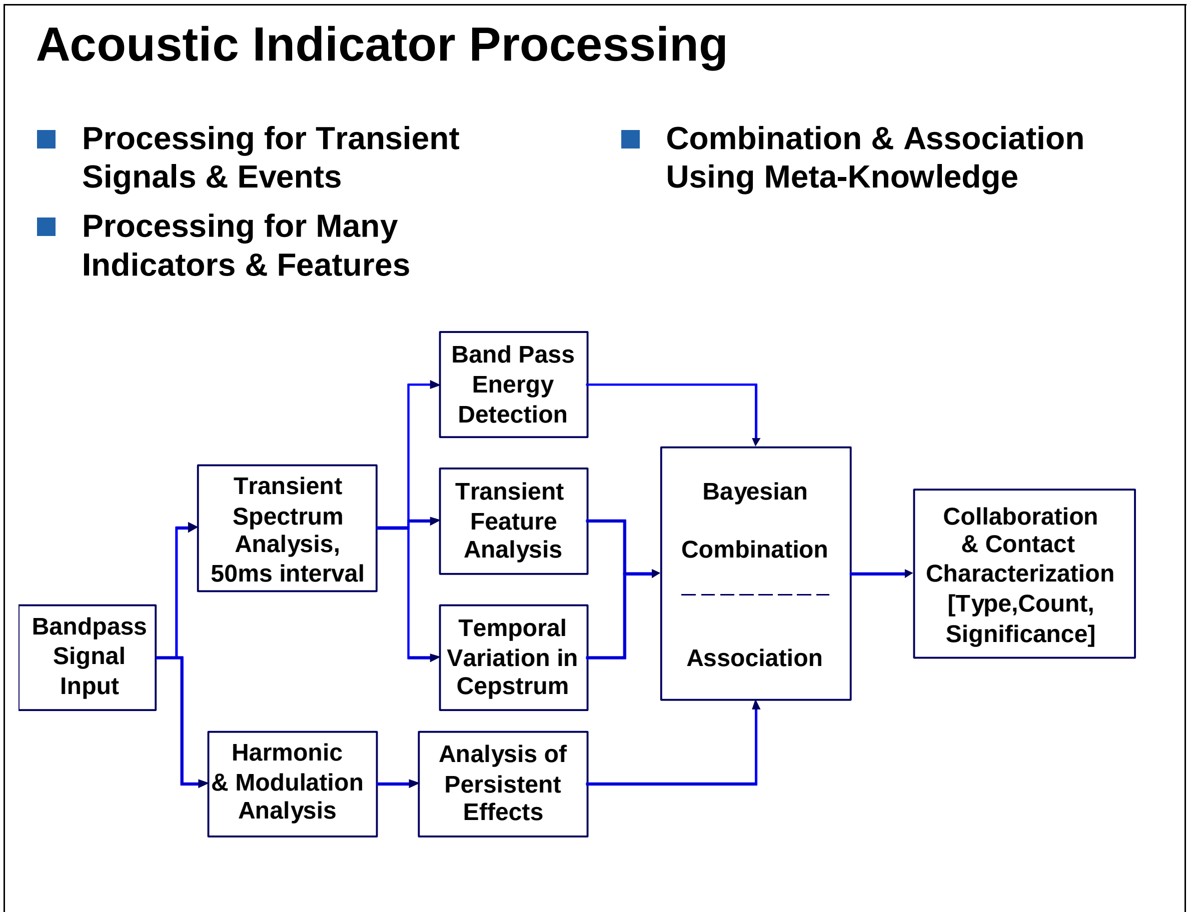


Figure 8-3 Acoustic Event Processing Flow

Seismic Indicator Processing

- Processing for Transient Signals & Events
- Processing for Many Indicators & Features
- Combination & Association Using Meta-Knowledge

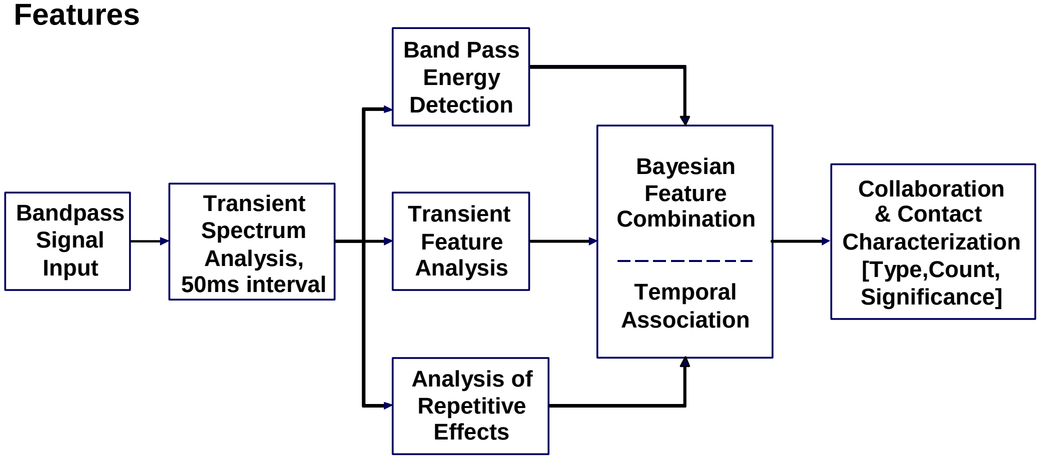


Figure 8-4 Seismic Event Processing Flow

Collaboration and Alert Generation

- Collaborative, Situation Sensitive, Alert Processing
- Effectively Utilizes Many Detection Features, Sequence, and Context to Produce Alarms

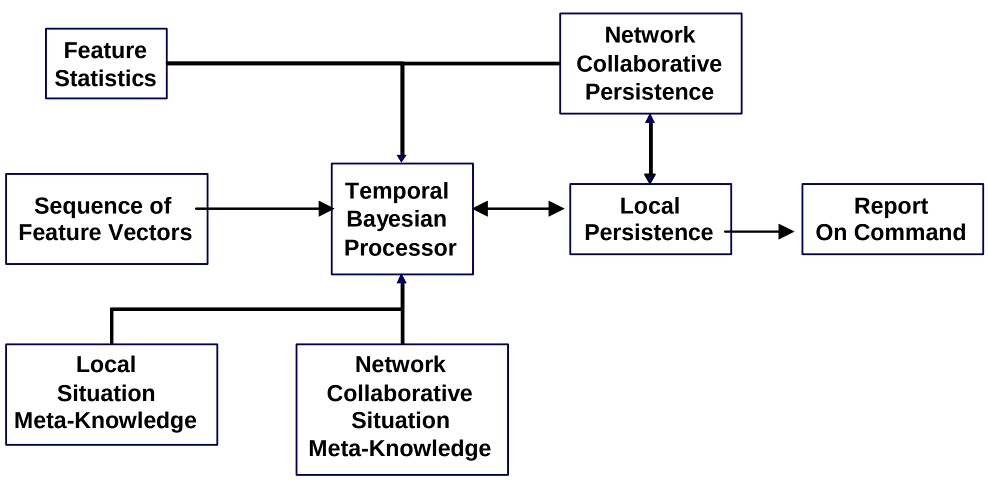


Figure 8-5 Sensor Fusion and Node Collaboration Processing Flow

9. SITEX00, JULY-AUGUST 2000

SensIT conducted demonstration experiments at the Marine Corps Air Ground Combat Center (MCAGCC), at Twentynine Palms CA during July and August of 2000. The site was in the eastern ranges at the south entrance to the Prospect as shown in Figure 9-1.

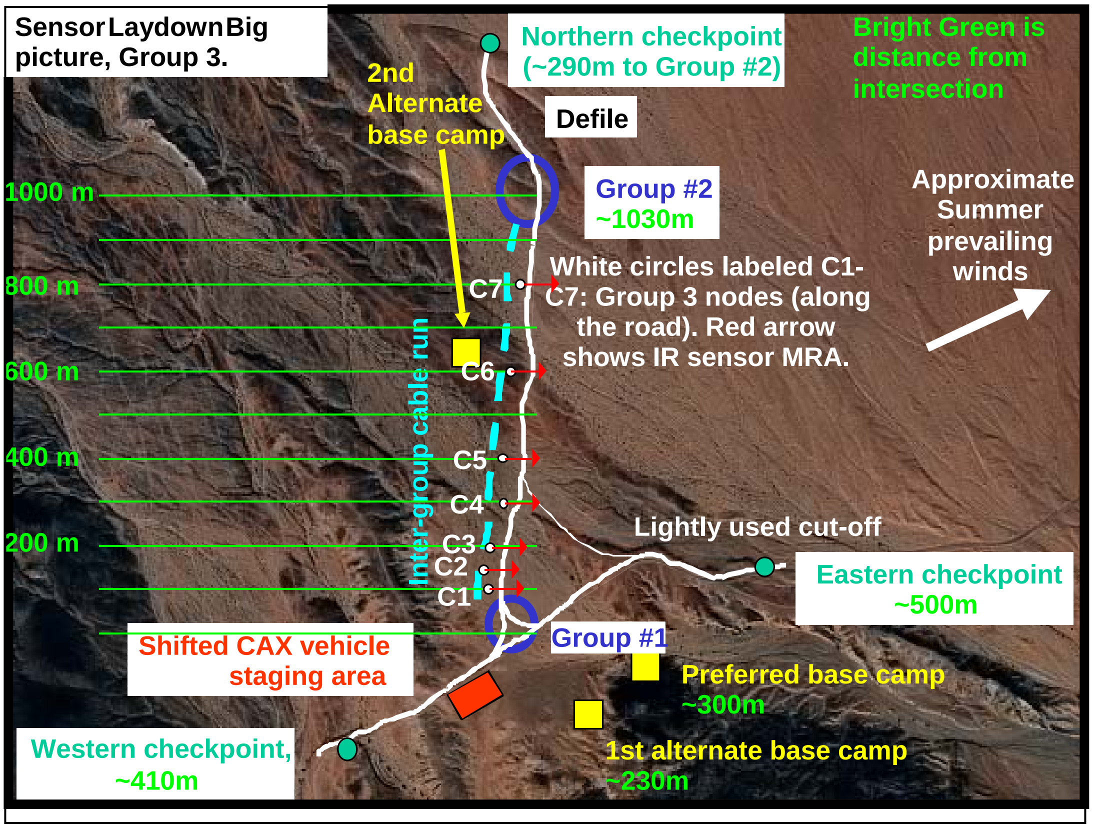


Figure 9-1 SITEX00 Experiment Laydown

BAE Systems supplied signal processing algorithms and software running on the Sensoria WINS 1.0 nodes for the exercise. The routines included time series signals, spectral signals, detection CPA events, and repositories.

BAE Systems extended their involvement past the nominal SITEX00 activity of signal collection using the WINS 1.0 nodes. Acoustic response on WINS 1.0 nodes was not sufficient to detect a vehicle horn at 10m. BAE Systems deployed a company developed acoustic data collection system to record high quality signals. BAE Systems also collaborated with ISI-W, MIT-LL, and PSU-ARL to integrate a collaborative vehicle detection application. The integrated system was tested on the last day of SITEX00.

BAE Systems Acoustic Data Collection System:

Several microphone arrays were deployed. They used GEXTEX 3307-5 electret microphones and pre-amp/line drivers to return signals over 300m lines to an IOtech 8-channel 12bit A/D, which was interfaced to a Dell laptop computer. Isolation transformers were used to control noise over the long distance lines. The laydown is shown in Figure 9-2 and Figure 9-3.

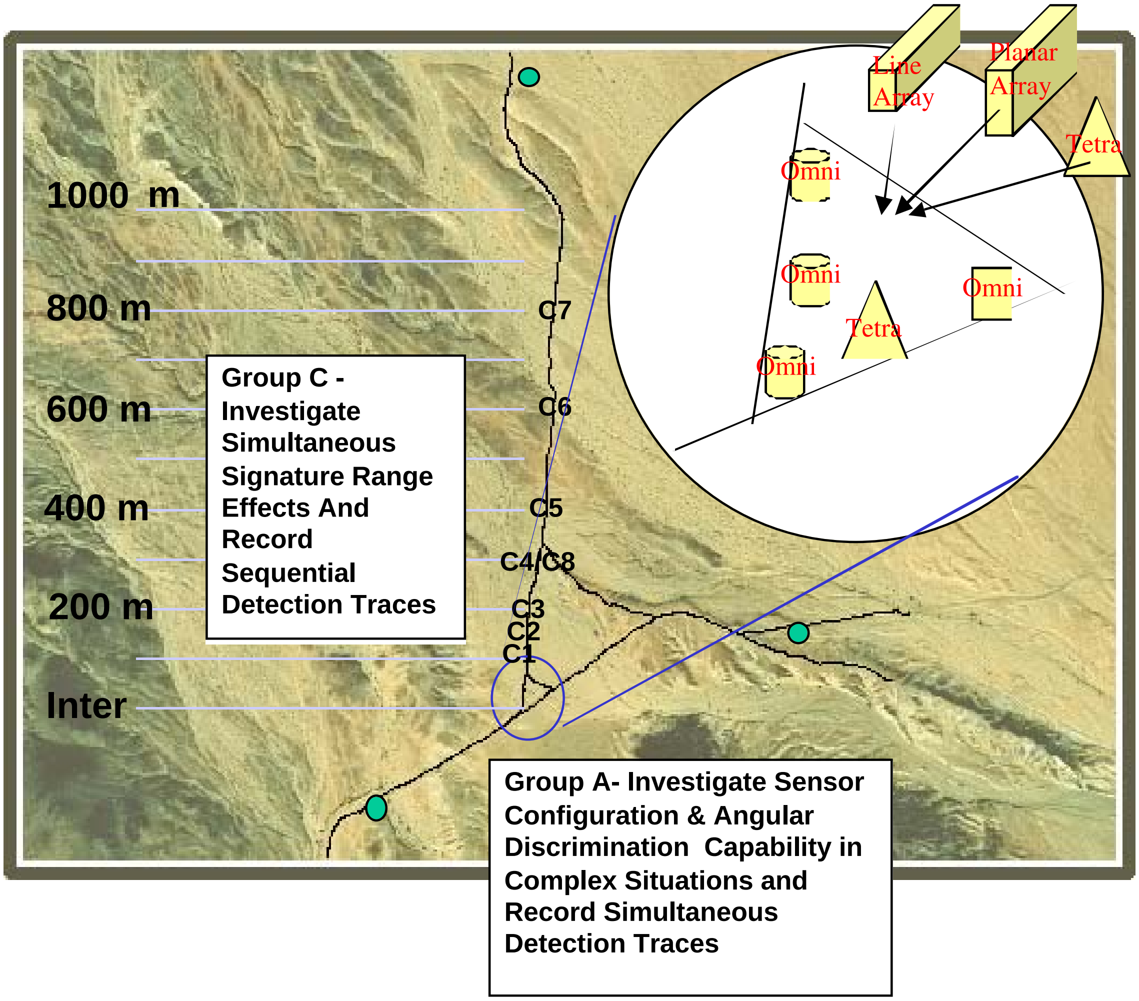


Figure 9-2 BAE Systems SITEX00 High Bandwidth Acoustic Sensor Lay Down

Installation was to secure unique scenario data and to investigate issues such as propagation effects. Eight omnidirectional microphones were deployed with variable spacing along the North-South road, starting with SensIT node C1 and extending 800m north to the defile. These nodes were serviced from a tent at node C4. They collected data for meso scale effect such as widely spaced signal coherence and propagation over a 50m to 150m distance. At node 4, two microphones were deployed; one at the height of Sitex00 nodes and one at 4” above the ground to investigate vertical propagation and ground heating effects. (Day time temperature at ground level reached 125F by 10am.)

Additionally, four omni microphones and four compact microphone arrays were deployed at the main road intersection. Two tetrahedron arrays, with 9cm element spacing, were deployed north and south of the intersection. An eight microphone line array, with 6cm element spacing, and a 2x4 microphone panel array, with 9cm spacing, were deployed next to the northern tetrahedron. These sensors were to investigate small scale effects such as closely spaced signal coherence and the utility of complex sensor configurations.

Wideband Sensor Locations

(as measured by a handheld GPS)

Sensor Locations Along the North-South Road

Sensor Description	Latitude	Longitude
C1	34'16'17.4	116'01'36.1
C2	34'16'19.0	116'01'36.4
C3	34'16'20.8	116'01'35.7
C4	34'16'24.0	116'01'35.0
C5	34'16'27.1	116'01'34.3
C6	34'16'30.4	116'01'34.3
C7	34'16'36.9	116'01'33.9
C8	34'16'24.0	116'01'35.0

Where C1 was closest to the intersection and C4 and C8 were at the same location. All sensors were approximately 1 meter off the ground with the exception of C8 which was 10 centimeters off the ground.

Sensor Locations at the Intersection

Sensor Description	Latitude	Longitude
L = Linear array	34'16'14.8	116'01'34.7
P = Planar array	34'16'14.8	116'01'34.7
Q = tetrahedron	34'16'13.7	116'01'34.7
R = tetrahedron	34'16'14.8	116'01'34.7
T9 = single omni acoustic	34'16'13.8	116'01'35.2
T10 = single omni acoustic	34'16'14.3	116'01'35.2
T11 = single omni acoustic	34'16'15.1	116'01'35.1
T12 = single omni acoustic	34'16'14.3	116'01'34.0

Figure 9-3 Wideband Sensor Locations

Scenes from the wideband data collection exercise are shown below in Figure 9-4. The cooler seen was used with ice and a fan to control the temperature for the computer and IOtech A/D. A typical day used 25lbs of cooling ice.

Figure 9-4 SITEX00 Wideband Data Collection Activity

Collaborative Detection System Integration on WINS 1.0 Nodes

Concurrent with the wideband data collection effort conducted at Twentynine Palms, the team of BAE Systems, ISI-W, MIT-LL and PSU-ARL worked at the ISI-W offices at Marina Del Ray, CA, to integrate a multi-node collaborative detection system, which was demonstrated at the end of Sitex00. For this effort, BAE Systems supplied signal processing and storage repositories to enable sharing for collaboration. PSU-ARL supplied multi-node decision algorithms. ISI-W and MIT-LL each supplied (competing) data communication routing algorithms to facilitate efficient localized communication among nodes. The integration extended over approximately one month and enabled comparison of two different versions of ‘Diffusion Routing’ algorithms, written to the same API over the preceding year by the ISI-W and MIT-LL teams. The integration effort salvaged year long software development work of all team members. It resulted in the discovery of a bug in the WINS 1.0 node radio code, and resulted in a successful experiment in multi-node collaboration. Integration activity at ISI-W offices is shown in Figure 9-5. A detection run at MCAGCC is shown in Figure 9-6.

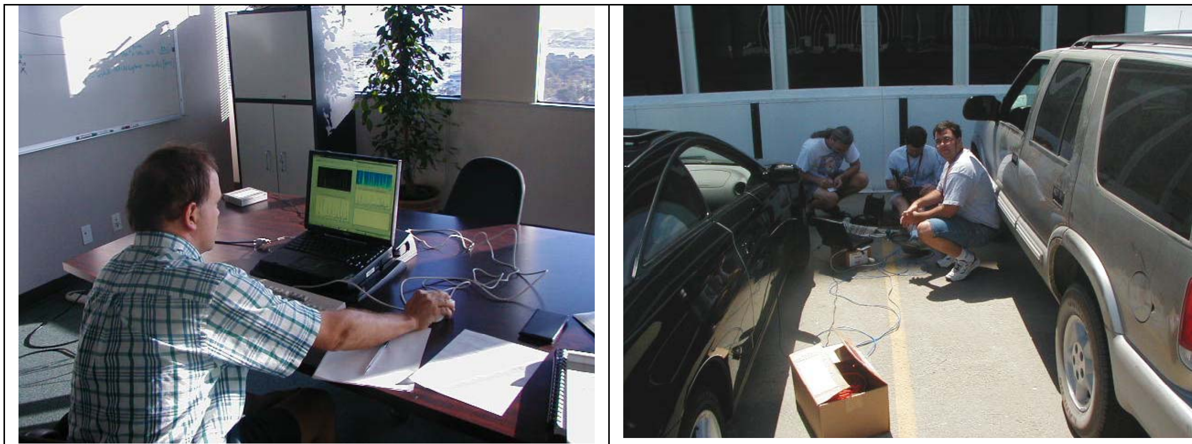


Figure 9-5 Scenes from Integration Effort

Figure 9-6 Collaborative Detection Target Run

10. SITEX01, MARCH 2001

BAE Systems participated in SITEX01, held at MCAGCC, 29 Palms, during March 8-15, 2001. Several teams demonstrated tracking applications and some signature data was collected on Soviet vehicles.

BAE Systems led a team that included PSU-ARL, ISI-W and Sensoria Corp. Sensoria supplied an imager compatible with WINS 1.0 nodes, an RF link to the SITEX01 command post, and a display. PSU-ARL supplied node–node collaboration interfaces to ISI-W supplied Diffusion Routing. BAE Systems supplied all signal processing, vehicle detection algorithms, Kalman trackers, and data repositories.

Building on the SITEX00 August 2000 experiment, the team jointly demonstrated a wireless imager triggered by a four node distributed tracker. Sensoria developed a wireless imager that can be externally triggered and can transmit a picture via a spread spectrum radio link to a laptop computer for display. BAE Systems developed robust CPA detection algorithms for seismic and PIR sensors, and a distributed Kalman tracking algorithm that ran on the WINS 1.0 nodes. The BAE Systems tracker transmits target state information among nodes using the ISI diffusion network routing, and calculates a target position and velocity estimate that is then used to trigger the imager. The experiment is indicated in Figure 10-1 and Figure 10-2.

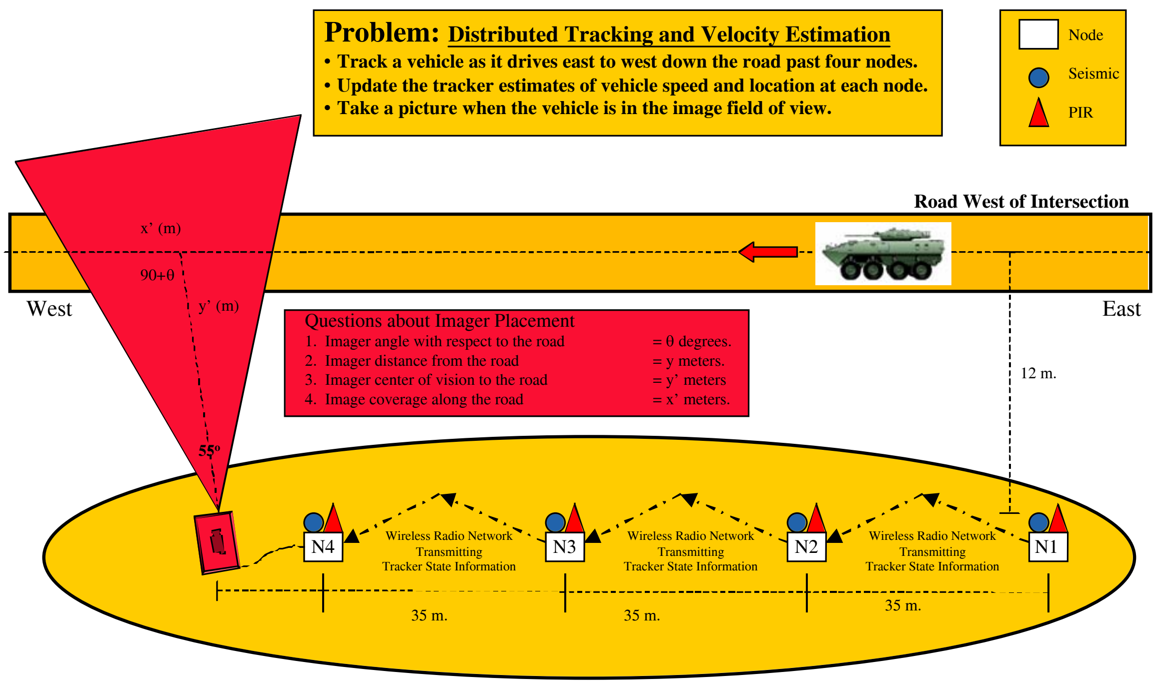


Figure 10-1 SITEX01 Tracking Experiment

Four WINS 1.0 nodes were located along a 100 meter stretch of road. These nodes self configured and used GPS to determine their location and to time synchronize. The first three nodes were used to detect a single vehicle traveling along the road, establish a direction and a track, and estimate the velocity. The detection and tracking processes are distributed across the network. Each node locally performs multi-sensor target detection and estimates the time of target CPA. Prior target state information, including target speed, direction, and track, is supplied from collaborating nodes, and will be updated on each node after a CPA event is detected. The updated target state information is transmitted over the wireless network, and shared as part of the low bit rate collaborative information vector. The fourth node monitored the target state information, and using node location information, estimated the time to trigger the imager. The detector uses a Neyman-Pearson optimum detection criterion and a robust CPA event detector with a 3-second detection latency. A simple AND fusion logic for seismic and PIR modalities is used to further reduce false alarms. Estimates of CPA target state information were displayed locally on each node screen. The transmitted image was displayed on the laptop screen.

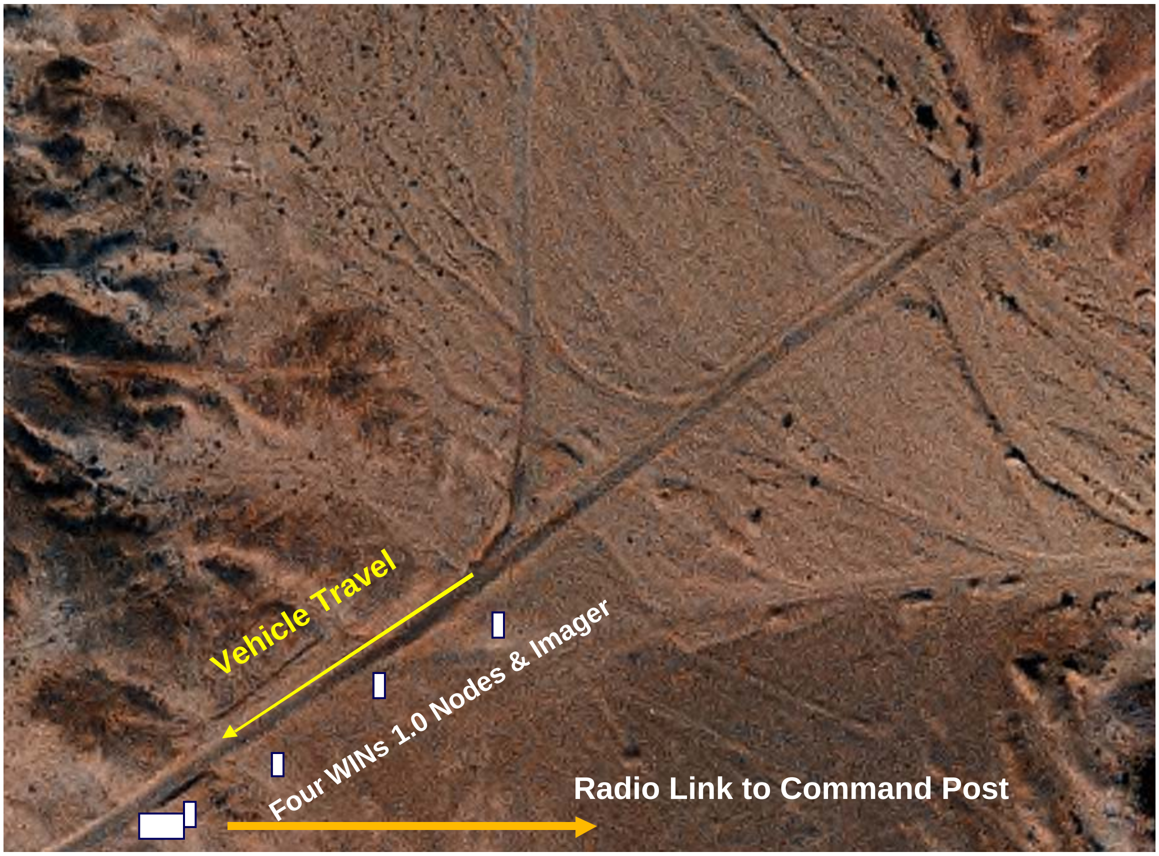


Figure 10-2 Tracking Experiment Lay Down.

The collaborative tracking system was designed to minimize data exchange. At initiation, nodes self configured and self located. They then broadcast their equipment configuration and location to others in the area which had subscribed to configuration exchanges. These simple exchanges allowed relative positions and internode distances to be calculated. During an event, the first node to detect a vehicle began a track and broadcast to the other nodes the Track ID and Time of CPA. This, with the locally stored distances, was sufficient for each node to update a local track copy. The node with the imager then used the track projection to trigger the imager when the vehicle was in the field of view. An overview of the processing is in Figure 10-3 and a sample image in Figure 10-4. A description of the Tracker API is in Figure 10-5.

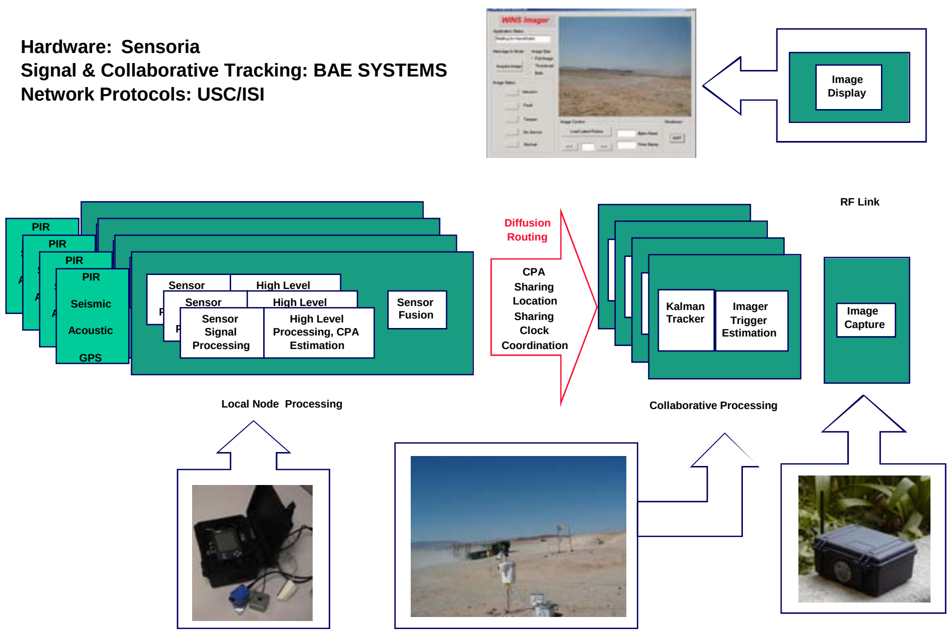


Figure 10-3 Collaborative Track Processing

Figure 10-4 Sample Imager Capture

Track point layout:
{ common information that can be handled by a GUI }
id string,{ id of track, arbitrary text }
time double,   { time of report, UTC, seconds since 1jan70 (linux system clock) }

{ target state information }
zone byte,   { UTM zone reference, -1 means unknown }
pnorth double,   { estimated location, meters north of reference point }
peast double,   { estimated location, meters east of reference point }
lconf float,   { location confidence [0.0, 1.0]}

vnorth double,   { estimated velocity in m/s, northerly component}
veast double,   { estimated velocity in m/s, easterly component }

class string,   { target classification}
cconf float,   { classification confidence, [0.0,1.0] }

{ tracker-specific information follows}
{ GUI will not understand without augmentation }
typecode string,   {specifies tracker-type}
trackspec blob   {tracker -specific fields – for example see below}

Notes:

This is not a record structure (yet).

This layout is called **Track point** because typically a sequence of such records, all with the same ID, makes up a target path (track). Thus, **id** is an id for a track, and each track point will carry that id for accumulating the track itself. All trackers must provide some sort of id for a track. The assumption will be that multiple track points with the same id are part of the same track.

Defaults and assumptions by GUI are:
time is the time a vehicle is at the estimated location. Default 0 (won't be displayed)
Zone, pnorth, peast will be valid UTM values and will be correct (i.e., in the zone the GUI is working). Default value of zone 0 means pnorth, peast are not reliable and there will be no display.
Vnorth, veast are estimates of speed of the target identified by zone, pnorth, peast, time. Default 0,0.
Class may be "Unknown"
Lconf, cconf have default values of 1.0.

---

example of trackspec for typecode = "Kalman"

Figure 10-5 BAE Systems Tracker API

SITEX01 was marginally successful. Vehicles were detected and tracked and an image captured. But use of GPS locations and time stamps for automatic node configuration proved problematical. Many otherwise successful runs were spoiled by improper time sync or by poor geo-reference; one instance the GPS reported the node to be located on the equator, with errors of 10km to 70km also noted. BAE Systems software did not account for the slow response of the WindowsCE operating system of the WINS 1.0 nodes and occasionally two copies of the program started and then competed for data samples.

The goal of low bandwidth collaborative tracking was proven. The short messages were completely adequate for distributed Kalman tracking. Lessons about slow data rates (the short messages were necessary or the radio couldn't send the message before the target passed the 'next' node), poor time synchronization, poor GPS interfaces, and a slow operating system all contributed to improved designs in the WINS 2.0 nodes.

11. SITEX02, NOVEMBER 2001

SITEX02 was the second SensIT field experiment involving all teams and was conducted from late October to mid November 2001. Different teams participated at different times to use the field of approximately 70 WINS 2.0 nodes supplied by Sensoria Corp. The nodes were located in the same general area as the nodes for SITEX00, at "the entrance to the Prospect" on the eastern ranges of MCAGCC, Twentynine Palms CA. As in SITEX00, all nodes were connected with 10BaseT Ethernet lines and powered from large lead acid batteries.

SITEX02 was a large exercise and several experiment teams participated. BAE Systems supported all teams with Signal Processing algorithms and Data Repositories for on-node data persistence. Two teams from Northwestern University separately examined collaborative detection and tracking. Rutgers and Berkley examined Mote/WINS integration. Auburn and University of Tennessee examined mobile services and target detection and classification. (This team continued the experiments in August of 2002 at the BAE Systems Austin Testbed.) Fantastic Data examined data caching for in-field persistence. ISI-W examined data latency and transport efficiency of their Diffusion Data Routing algorithm. MIT-LL examined multi-node localization with TDOA techniques. Xerox PARC examined multi-node tracking and use of acoustic arrays for localization. A large team headed by PSU-ARL, including University of Maryland and Virginia Tech, examined a distributed vehicle tracker. BBN joined this team at the end of SITEX02 with a substitute tracker. A separate BAE Systems (Nashua) team operated MIUGS equipment during the SITEX02 runs. BAE Systems (Austin) also operated high bandwidth acoustic collection systems and examined other sensors, including magnetometers and microradar.

BAE Systems developed new versions of the signal processing algorithms and new versions of the data repositories. Work started with the WINS 2.0 specification published after the April PI meeting. New code was tested and documented between the delivery of the first WINS 2.0 nodes in early August and the systems integration conference at the first of September. Algorithm documentation, API documentation and early library iterations were supplied to other teams during July and August to support their development needs. Telephone consulting and e-mail exchanges aided the users. BAE Systems supported the September integration meeting conducted at BBN by delivering the proven signal processing algorithms and data repositories as tested libraries re-implemented for the LINUX operating system of the WINS 2.0 nodes. Brian Corser attended the integration meeting to provide training and software support to the users. During field integration and test periods, BAE Systems aided other teams with unique interface code and with design and debug support.

BAE Systems interfaced to the WINS 2.0 signal capture and GPS sub-processor. Acoustic, seismic, and PIR motion sensors were fitted to all nodes for the group experiments. BAE Systems published properly down sampled time series in the Time Series Repository for shared access by all teams. Power spectra and filtered bands of interest were published in the Signal Processing Repository. CPA detections, with times and confidence, for all sensor modes were published in the High Level Event Repository. Node location and time synchronization information, generated with MIT-LL support, were published in the Meta Knowledge

Repository. BAE Systems generated detection events became the default system health indicator – intentional traffic and targets-of-opportunity generated events and failure to detect was a reliable sign of node failure.

During the field exercises at MCAGCC, BAE Systems focused most of their effort on signature collection with the wideband data collection system. Having delivered required algorithms at the September integration meeting, only occasional support to teams using the algorithms was required. Data collection was conducted from a tent located just north of the main road intersection. A second BAE Systems tent was installed to support field integration teams from Auburn, Xerox PARC, and MIT-LL.

BAE Systems installed a buried magnetometer under the middle of the North-South road and a second magnetometer at 15m from road center. A 3-axis reference geophone was installed in the group of instruments 15m from road center. Several accelerometers attached to surrogate sensor node bodies were installed to investigate accelerometer versus geophone seismic sensing. The microradar was installed at the 15m roadside cluster. And, an omni-directional microphone was installed at this same site. See Figure 11-1 for an overview of the sensing cluster that was adjacent to one of the BBN sited WINS 2.0 nodes.

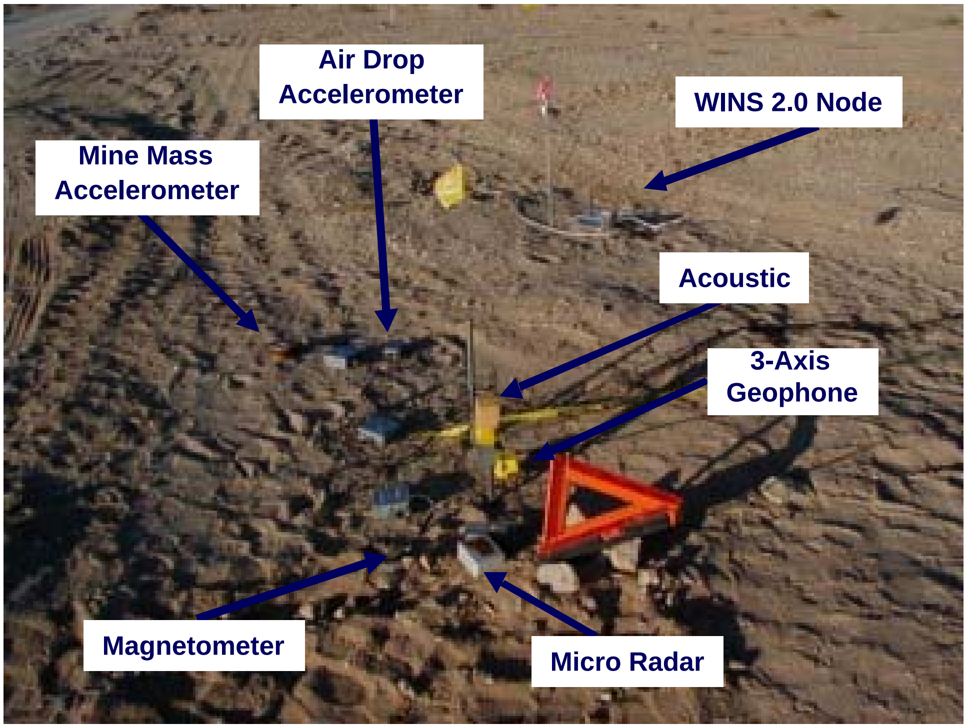


Figure 11-1 BAE Systems Road Side Instruments for SITEX02

Additionally, some acoustic arrays were installed. The line array, panel array, and tetrahedron arrays from SITEX00 were installed approximately 40m from the North-South Road. A mesoscale array of seven omnidirectional microphones was installed south of the BAE Systems operational tent and the road intersection. The separation of these microphones was to approximate the placement of mines deployed by a Volcano launcher; approximately 15m displacement. Their pattern was to facilitate the study of possible multi-node beam forming or coherent processing on this intermediate scale. These installations are in Figure 11-2. Data from these sensors were collected using the IOtech and Laptop Computer data acquisition from SITEX00. Files were shared with other SensIT PIs as requested.

Figure 11-2 BAE Systems Acoustic Arrays at SITEX02

12. BAE TESTBED IN-BUILDING TRACKING, July 2002

While most SensIT effort was focused on vehicle problems, BAE Systems investigated sensing in urban and complex terrain. Algorithms for human presence detection were implemented on WINS 2.0 nodes positioned in and around buildings on the Austin campus. The Kalman Tracker used in SITEX01 at MCAGCC, Twentynine Palms, was modified for tracking humans as they moved about the building. The SensIT team from ISI-E joined in a series of experiments to monitor the Austin building site remotely from a center just outside the DARPA offices.

A building defense scenario was investigated. In this scenario, a small force is inside a building to keep it secure. A vehicle with an assault team approaches the building, troops enter the ground floor through a door; some go to the second floor to take a corner window position and provide cover fire for any external defense response team. Sensors are placed to monitor this and alert the Blue force of conditions. A node in the approach area detects and identifies the troop vehicle and provides the initial alert. A motion sensor at the door counts troops entering the building. A motion sensor counts troops as they arrive at the second floor, giving the Blue team a measure of strength on each floor. The time to move from the door to the second floor indicates intent; a long delay indicates a thorough clearing of the first floor. On the second floor, sensors monitor movement and from track evidence indicates Red troop locality and control of hallways. The rapid movement to the corner indicates that this splinter force intends to focus outside and let the following force clear the remainder of the building.

BAE Systems tracker allowed person-track association under the erratic movement of persons on a mission. Detections and track activity was relayed to an ISI-E server for display at DARPA. WebCams, also linked to the server, provided ground truth about movement and hallway control. This experiment was replicated at the final PI meeting at BBN, November of 2002, with the addition of the Cornell University team, who supplied node and network query capability and mobile code. A sensor laydown and site pictorial are shown in Figure 12-1 and Figure 12-2.

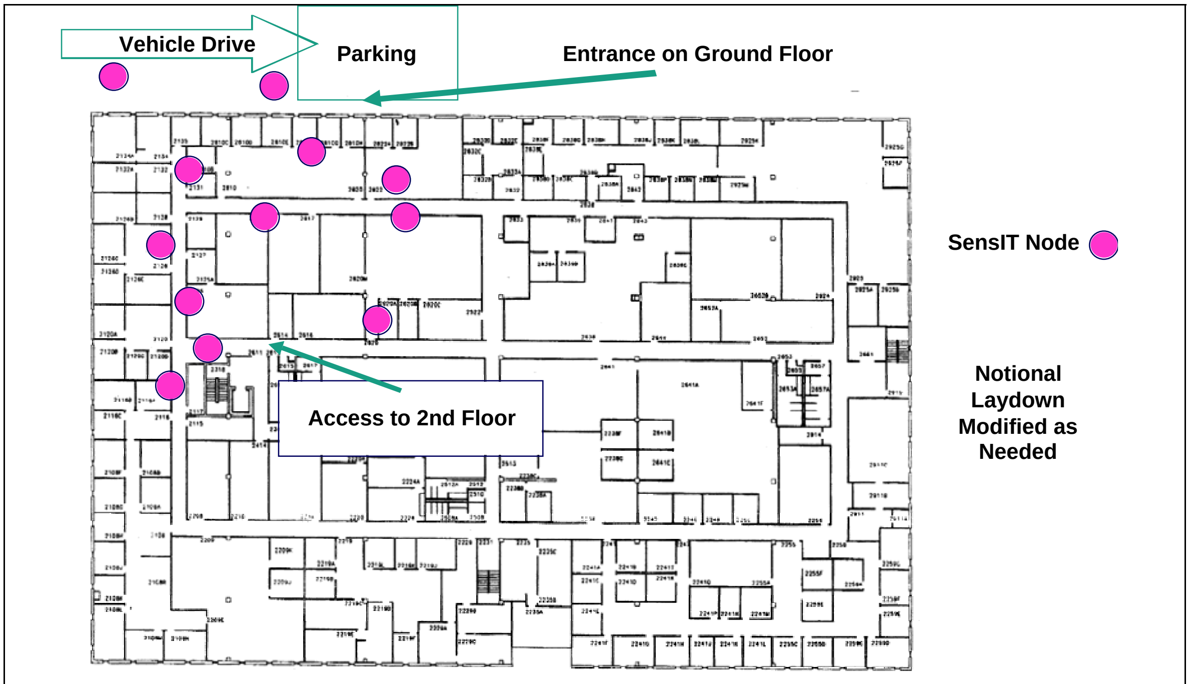


Figure 12-1 BAE Systems In-Building Experiment Lay Down

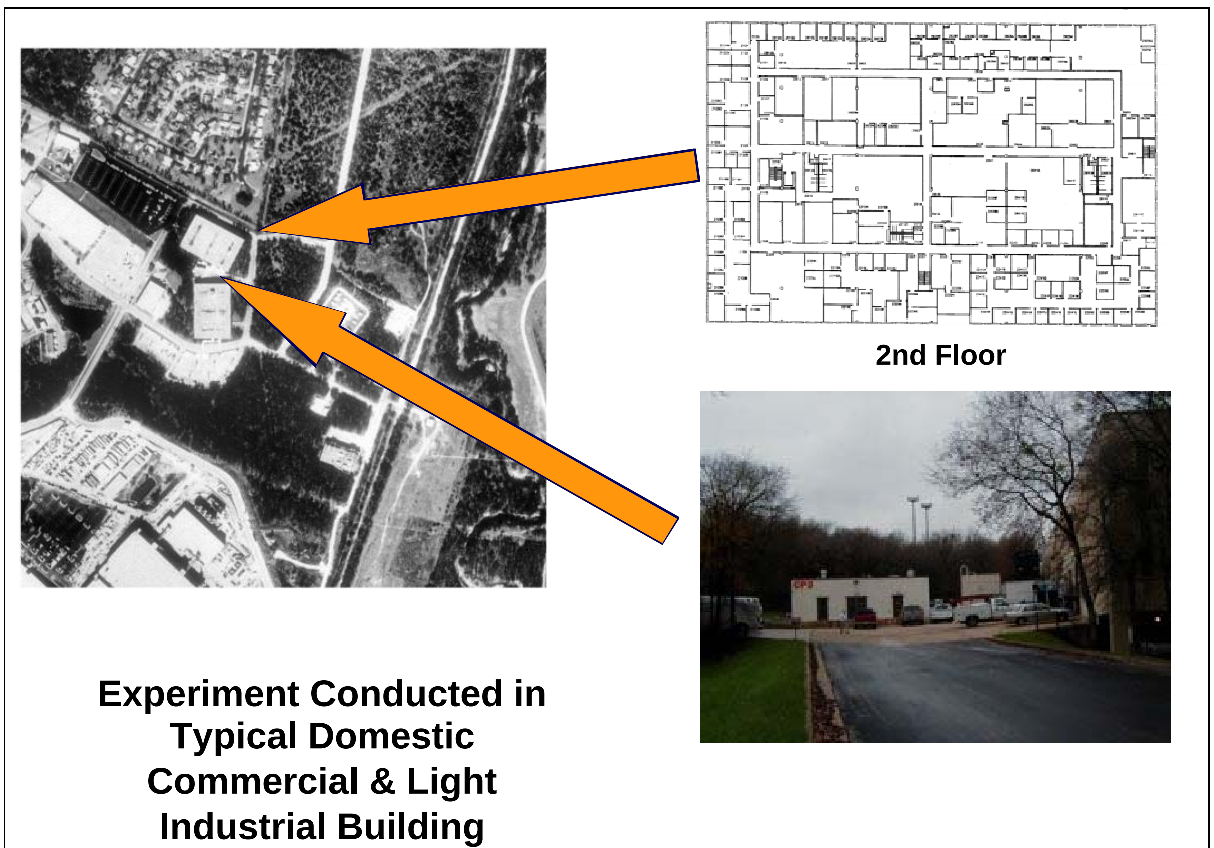


Figure 12-2 BAE Systems Urban Test Site

13. BAE TESTBED OUTDOOR TRACKING, AUGUST 2002

BAE Systems installed a SensIT Testbed of 22 WINS 2.0 sensor nodes during the summer of 2002. A variety of conditions were available on the Austin Campus as seen in Figure 13-1 and Figure 13-2.

The SensIT WINS 2.0 nodes were mounted in a weather proof case with a 120AH Marine Battery. Each node was configured with both 802.11 wireless and 10BaseT wired network connections. The internet links were to support node configuration, software management and run time monitoring, raw time series data collection, and links to the Campus Internet for links to off-site experimenters. A node is shown in Figure 13-3.

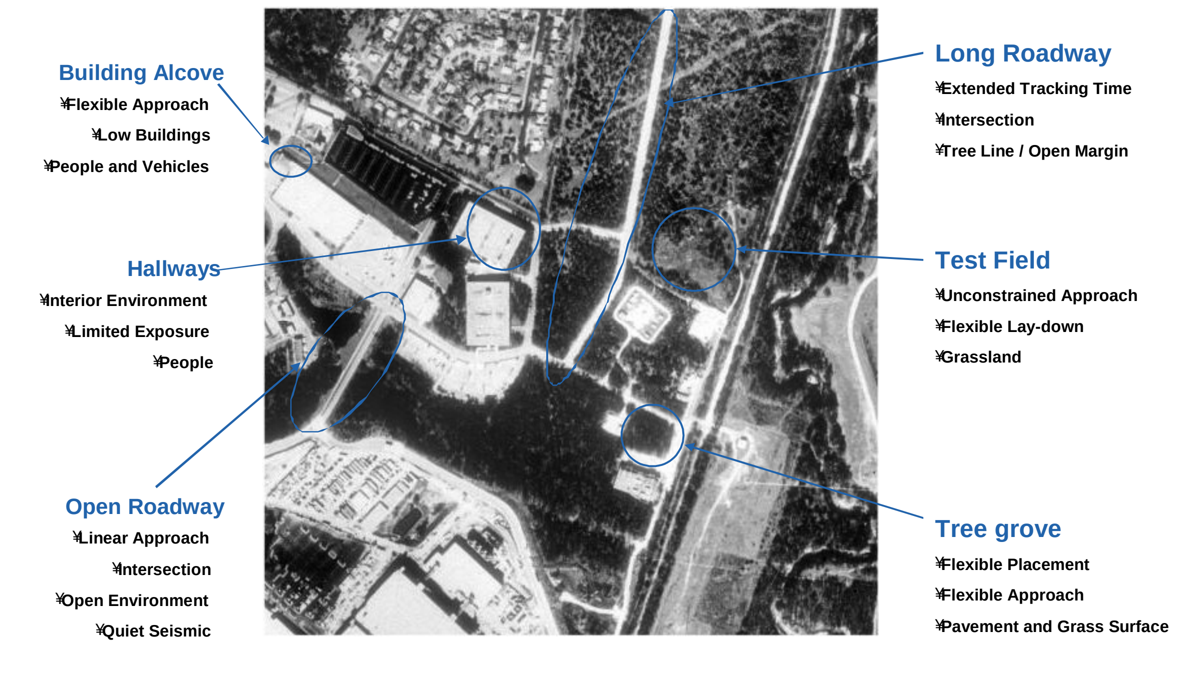


Figure 13-1 BAE Systems Austin Test Bed Experiment Locations

Figure 13-2 BAE Systems Experiment Environments

Figure 13-3 BAE Systems Test Bed Node Configuration

A multi team experiment was conducted on the test bed during August 17-22, 2002. BAE Systems provided signal processing and repository services on the WINS 2.0 nodes. ISI-W Diffusion Routing supported node-node collaboration and data exchange. Auburn University provided versions of target detection and Dynamic Software Services, which support sensor field re-configuration in the event of failure. University of Tennessee conducted experiments in target detection, tracking, and identification.

Two sites were used: a roadside intersection with structured sensor placement, working against a variety of planned and targets of opportunity; and a 'randomly' placed configuration in a open field (a parking lot).

Software and experiment site integration began over the weekend of August 17. Experiments ran Monday – Thursday. Most time was spent at the road intersection, as it was convenient and supplied a steady stream of targets. The parking lot was used on Wednesday, with SUV and heavy pick-up truck targets. During these runs, most all of the SensIT Challenge Configurations were accomplished. Photographs of experiment activity are shown in Figures 13-4 and Figure 13-5.

Figure 13-4 BAE Systems Test Bed Experiment Roadside Activity

Figure 13-5 BAE Systems Test Bed Experiment Parking Lot Activity

14. CONCLUSION

SensIT promoted a series of advancements in the understanding and use of fields of autonomous, collaborative sensor nodes. Two node design iterations were completed and follow-on designs specific to team members were sponsored (BAE Systems developed a very small low power node to detect human presence). The utility and efficiency of Diffusion Routing was demonstrated and several version iterations delivered during the project. Data persistence was examined; both on-node storage with share-on-demand, and cached storage with predictive sharing, being studied. Several experiments in collaborative sensing, collaborative localization, and collaborative identification were performed by a long list of teams. A distributed collaborative tracker was demonstrated. Self configuring nodes, supported by a self forming radio network, were demonstrated. Configurations with structured laydown, and dense and randomly placed nodes were investigated. BAE Systems supported all these with publicly shared signal processing and repository libraries of the highest quality.

The body of technology accomplishments described above establishes a technology base for the transition of sensor networks to use in modern battlefield settings like Afghanistan and Iraq. BAE Systems has continued to develop and mature the SensIT technology base. We have focused on developing robust sensor nodes that can be deployed to form an autonomous self forming sensor network. Figure 14.1 shows the sensor node being supplied to the Pathfinder ACTD program sponsored by the Special Operations Command (SOCOM). Active discussions are also underway with the Marine Corp and Air Force to supply sensor networks for border and remote airfield monitoring. Situational awareness in urban environments is also a priority that is being pursued by BAE Systems. Low power autonomous sensor networks are key to success in the tactical urban environment. In summary, SensIT demonstrated that large numbers of sensors can be deployed in a self forming ad hoc wireless sensor network to provide effective battlefield sensing. It is an ongoing effort to transition this technology for service use.

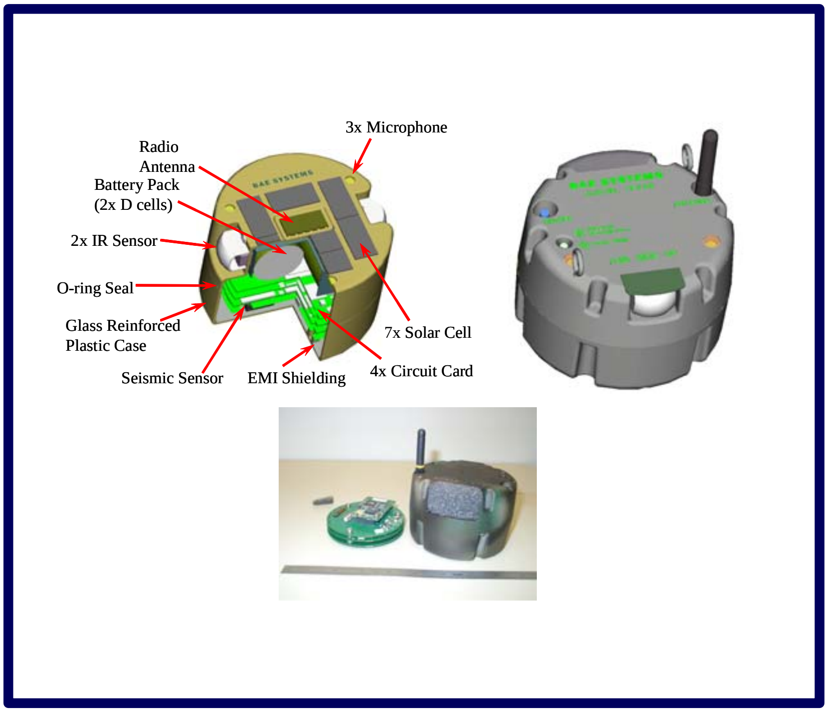


Figure 14.1 BAE Systems Wireless Sensor Node

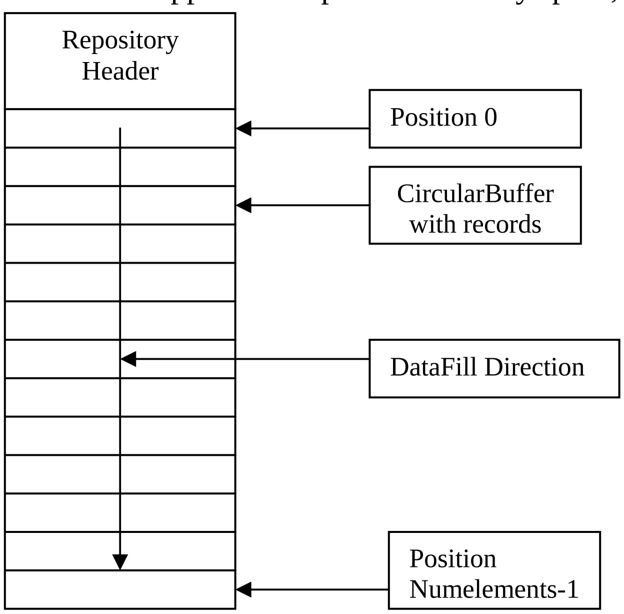
APPENDIX A: BAE Repositories API

Overview

The purpose of this document is to provide a description of the repository architecture along with an explanation of the API. Example code will also be presented that demonstrates how to add data to a repository, retrieve data from a repository and receive notifications when new data has been added a repository. Some of this is out of date but most of the function calls and architecture are still the same.

Repository Architecture

The repositories were designed to provide an efficient means of storing and retrieving real time data that is shared between several independent processes. After a fair amount of research, it was determined that memory mapped files with advisory record locking provided the most efficient form of Inter Process Communication (IPC). This is in light of the fact that at the current time IPC V is not supported on the SH4 linux boxes. The binary layout of a repository, once it has been mapped into a process memory space, is shown below.



The repository header is a structure that is described below and is defined in RepUtil.h.

Field Name	Type	Description
Name	char[128]	Name of the repository
Numelements	Int	Number of elements the circular buffer holds
Elesize	Int	Record size in bytes
Poshead	Int	Zero based position index of the newest record in the buffer
Postail	Int	Zero based position index of the oldest record in the buffer
Flagfull	Int	Flag that indicates if the circular buffer has been filled yet and wrapped back around

One file will be created and mapped for each repository. The naming convention for these files is Rep#.rep.

In order to interface with the repositories, a small static link library has been written by BAE SYTEMS. This library provides the following functionality.

1. Allows a user to connect to a repository, or automatically create a newly initialized one if there are currently no users.
2. Add data to a repository
3. Get the repository header
4. Retrieve data from a repository
5. Get the a copy of the newest record
6. Get the position of the newest record

These functions are described in the appendix.

Subscription Manager

The subscription manager allows users to subscribe to different repositories so that they can be notified when new data is available. The first time a user enters a subscription, the following events take place.

1. A fifo is created that is used to receive event notifications.
2. A thread is created to service the fifo
3. The subscription is written to Subscriptions.rep

If the subscription is changed at a future point in time, only the file entry is updated. Currently, the number of subscriptions is limited to 100.

When data is added to a repository that a program has subscribed to, an event structure will be created and then placed in the program's fifo. The table below describes an event structure, which is also declared in RepUtil.h.

Field Name	Type	Description
Timestamp	Timeval	Time that the event was placed in the fifo
Repid	REP_ID (enumeration)	Id of the repository
Pos	Int	Position within the circular buffer the record was placed

Event structures are then passed to the user via the callback function. The timestamp value is set when the event is written into the fifo with the gettimeofday function, and therefore may not correspond to the timestamp on the record. Also, another thing to note is that if a program's processing cannot keep up with the rate at which events are being generated, then its event queue will eventually fill and events will be lost. Currently, the event queue can hold 100 event structures.

In order to avoid unexpected program termination the repositories setup a signal handler to handle the SIGPIPE signal. If your program also uses SIGPIPE then you should contact BAE so that special provisions can be made.

Repository API

This section will describe the functions available for interfacing with the repositories and are defined in RepUtil.h.

void InitRepository()

Description: This function should be the first repository function called. It is used to initialize various data structures.

Inputs: None.

Outputs: None.

REP_RESULT CreateRepository()

Description:
This function is used to create the repositories files and map them into the user's memory space. The repository is created in the current directory. If this call detects that there is a repository already in use, it will just connect to it. The file RepLockFile.rep is used by this function. Currently, this function creates eight repositories; four time series and four signal processing. The time series repositories all hold the TS_RECORD structure and the signal processing all hold an SP_RECORD structure; both are defined in RepUtil.h.

Inputs: None.

Output: REP_OK on success.

REP_RESULT RepAddData(REP_ID id,void *pdata, int size)

Description:
This function is used to add new data to the repository. Once the repository in full the oldest record will be overwritten when this function is called.

Inputs:
id - One of the repositories enumerated in REP_ID
pdata - A pointer to the record that is to be copied into the repository.
size - The size of the memory pointed to by pdata. REP_INVALID_RECORD_SIZE is returned if size does not match the size(elesize) of the record.

Outputs:
REP_OK on success.

REP_RESULT RepGetHeader(REP_ID id, REP_HEADER *phdr)

Description:
This function is used to get a copy of the repository header. Poshead and Postail will be equal to –1 if no data has been added to the repository.

Inputs:
id - One of the repositories enumerated in REP_ID
phdr - Pointer to a REP_HEADER that will receive the data.

Outputs:
REP_OK on success.

REP_RESULT RepCopyDataAbs(REP_ID id, int index, void *pbuf, int size)

Description:
This function is used to provide random access to repository data.

Inputs:
id - One of the repositories enumerated in REP_ID
index - Position in the circular buffer to retrieve data from.
pbuf - Point to the memory that will hold the record.
size - Size of the memory in bytes. This should be the same size as the record.

Outputs:
REP_OK on success.
REP_INDEX_OUT_OF_RANGE – If the index is less than 0 or greater than numelements-1
REP_NO_VALID_DATA – This value is returned if no data has been added at this position.

REP_RESULT GetNewestDataPos(REP_ID id,int *index)

Description:
This function returns the position of the newest data record, which would be useful, if an application were going to use polling to determine when new data was added to the repository. When new data is added the index will change.

Inputs:
id - One of the repositories enumerated in REP_ID
index - Position in the circular buffer to retrieve data from.

Outputs:
REP_OK on success.

REP_RESULT GetNewestRecord(REP_ID id, void *pdata, int size)

Description:
This function is used to get a copy of the most recent record in the repository.

Inputs:
id - One of the repositories enumerated in REP_ID
pdata - Pointer to the memory that record will be copied to.
size - Size of the memory pointed to by pdata.

Outputs:
REP_OK on success.

REP_RESULT SetSubscriptionManager(CSubscriptionManager *psm)

Description:
This function is used to set the subscription manager that the repository is going to use. If the program does not want to notify other users when new data is available or receive events then it should not call this function. Care should be taken to initialize the subscription manager before calling this function.

Inputs:
psm - A pointer to an initialized subscription manager.

Outupts:
REP_OK on success.

CSubscription Manager API

This section will describe the CSubscription manger class Api. The class is declared in SubscriptionManager.h and if a program uses the subscription manager, it should include this header file after RepUtil.h.

REP_RESULT Init(char *name,void *(func)(REP_EVENT))

Description:
This function is used to initialize the subscription manager. It should be called after the InitRepository() and CreateRepository() function calls and before the SetSubscriptionManager call.

Inputs:
name - A character pointer to unique string that will be used to identify the user. Only the first 24 characters will be used.
func - A pointer to a user function that the subscription manager will call when new data is available. If the user does not want to receive notifications NULL should be passed in. This would allow a program to still broadcast notification to other programs.

Outputs:
REP_OK on success.

REP_RESULT SetSubscription(CSublist *list)

Description:
This function is used to set the repositories that the user wants to subscribe to.

Inputs:
list - A pointer to a CSublist object that contains the subscription list.

Outputs:
REP_OK on success.

REP_RESULT DeleteSubscription()

Description:
This function deletes the current subscription from the repository.

Inputs:
None.

Outputs:
REP_OK on success.

Example Code

This section will present two short examples that should help to demonstrate the API and the calling order. The first example is a simple server that just puts data into time series repository one. The second example is a client program that sets up a subscription for time series one. To build the example programs, copy the source code into a directory and type the following commands:

make –f replib.mk Replib
make –f replib.mk Examples

The first command will build the static link library RepLib.a and the second command will build ExampleServer and ExampleClient. To run the example programs, type ./ExampleServer to start the server. Then in a separate console window type ./ExampleClient xxx. The ExampleClient program takes a command line argument, which is the name of the subscription. More than one ExampleClient can be run at a time, but they will need to be started with different subscription names. The code below gives an example.

start the server
>./ExampleServer
separate console window
>./ExampleClient client1
separate console window
>./ExampleClient client2

ExampleServer

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include "RepUtil.h"
#include "SubscriptionManager.h" // include afer RepUtil.h

int main()
{
	CSubscriptionManager sm;
	REP_RESULT		res;
	TS_RECORD		tsrec;
	int			i;

	// initialize and create the repositories
	// these should always be the first two calls
	InitRepository();
	res = CreateRepository();
	if(res != REP_OK)
	{
		printf("Error creating repository\n");
		exit(1);
	}

	// We can now initialize the subscription manager
	// Null means that we are not interested in
	// receiving callbacks
	res = sm.Init("ExampleServer",NULL);
	if(res != REP_OK)
	{
		printf("Init error\n");
		exit(1);
	}

	// set the subscription manager so that we can broadcast
	// events to other users
	SetSubscriptionManager(&sm);

	i = 0;
	for(;;)
	{
		tsrec.data[0]   = i;
		tsrec.data[255] = i;

		printf("%d\n",i);

		i++;

		// add a new record to the time series repository
		// this function will also notify other interested users
		// since we attached a subscription manager
		res = RepAddData(TS_CHAN_1,&tsrec,sizeof(tsrec));
		if(res != REP_OK)
		{
			printf("Error adding data\n");
			exit(1);
		}
		usleep(0);
	}
}
```

ExampleClient

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include "RepUtil.h"
#include "SubscriptionManager.h" // include afer RepUtil.h

void RepCallback(REP_EVENT ev);

int main(int argc,char *argv[])
{
	CSubscriptionManager sm;
	REP_RESULT	res;
	TS_RECORD	tsrec;
	CSubList	sl;
	int		i;
	char		repname[25] = "Test_Rep"; // default user name

	if(argc > 1)
	{
		strncpy(repname,argv[1],25);
		repname[24] = '\0';
	 }

	// initialize and create the repositories
	// these should always be the first to calls
	InitRepository();
	res = CreateRepository();
	if(res != REP_OK)
	{
		printf("Error creating repository\n");
		exit(1);
	}

	// We can now initialize the subscription manager
	// repname is passed in via the command line and
	// RepCallback is a pointer our callback function
	res = sm.Init(repname,RepCallback);
	if(res != REP_OK)
	{
		printf("Init error\n");
		exit(1);
	}

	// set the subscription manager so that we can broadcast
	// events to other users
	SetSubscriptionManager(&sm);

	// Subscribe to time series channel 1
	sl.Subscribe(TS_CHAN_1);

	// Set the subscription so that we can start receiving callbacks
	res = sm.SetSubscription(&sl);
```

```
	if(res != REP_OK)
	{
		printf("Set subscription error\n");
		exit(1);
	}

	i = 0;
	for(;;)
	{
		sleep(10);
	}
}

void RepCallback(REP_EVENT ev)
{
	static TS_RECORD tsrec;

	RepCopyDataAbs(ev.repid,ev.pos,&tsrec,sizeof(tsrec));
	printf("rep id %d pos %d data[0] %d data[255] %d\n",ev.repid,
					ev.pos,
					tsrec.data[0],
					tsrec.data[255]);
}
```

APPENDIX B: Example Repository Polling Code

Overview

The purpose of the document is to provide documentation for the example code that is used to poll the repositories for data. It will also explain how to run and configure the signal processing code that puts data into the repositories.

Repository Code

Several changes and additions have been made to the repository code since its very first release. One of the most notable changes is that structures have been defined to support one of three different types of processing (i.e. Acoustic, Seismic, and PIR). A high level repository has also been added and a structure for it has also been defined. These structures can be found in the ProcStruct.h header file. The header file is also being included at the end of this document. A file call config.rep is used to configure the repositories. It is read in when the call to CreateRepositories() is made. If this file is not found in the current directory, a default configuration is used and a config.rep file is generated that reflects the current settings.

Several new functions have also been added. Among these are functions that allow a type safe way to add and retrieve data from the repositories. These functions are in RepTypeSafe.h and RepTypeSafe.cpp. All of these functions follow the same basic form. An example of one of these function is given below:

```
void RepAddAcousticTs(REP_ID id,AcousticSection *prec);
```

This function allows a user to add data to the repository. If the repository identified by id has not been setup to hold an AcousticSection, then an error message will be printed and the program will exit. These functions basically wrap the RepAddData and the RepCopyDataAbs, except that they check to see if the repository has been setup correctly. Another new function that has been added is:

```
REP_RESULT RepLookupRecord(REP_ID id,timeval rectime,int *index)
```

The purpose of the function is to retrieve a record's position based on a time stamp. If rectime does not fall in the time span of the data being held in the repository, then either REP_TIME_OUT_OF_RANGE_LESS or REP_TIME_OUT_OF_RANGE_GREATER will be returned depending of whether rectime is less the oldest record time or greater then the newest record time in the repository.

REP_ID is defined by the following enumeration:

```
{
        TS_CHAN_1, // time series repository for a/d channel one
        SP_CHAN_1, // low level signal processing repository for a/d channel one
```

```
        TS_CHAN_2, // time series repository for a/d channel two
        SP_CHAN_2, // low level signal processing repository for a/d channel two

        TS_CHAN_3, // time series repository for a/d channel three
        SP_CHAN_3, // low level signal processing repository for a/d channel three

        TS_CHAN_4, // time series repository for a/d channel four
        SP_CHAN_4, // low level signal processing repository for a/d channel four

        HIGH_LEVEL_REP, // holds detection results

        NUMBER_OF_REPS,
};
```

Another significant change to the repositories is that a program that uses them can now be run from an nfs mount drive. The reason this was not possible in the first release is that the repository files where written to the current directory and linux does not support file locking on network files. To overcome this problem we are now setting the repositories up in /tmp.

Signal Processing

The signal processing is contained in a program call sh4baeproc and should be in the /bin directory of the zipped file. This program does the low-level processing, detection, classification and writes the results to the repository. It can be configured to perform one of four types of processing Acoustic, Seismic, PIR and No Processing. The type of processing performed and gain for each channel is determined by the config.rep file. This program also needs the following files: HPFfilcoefs.dat, Hanning1024.dat, Hanning256.dat and dsfiltcefs64.dat.

config.rep

config.rep is used to configure both the repositories and sh4baeproc. An example config.rep file is shown below.

```
####################################################################################
#
# BAE Repository Configuration File
#
# lines that start with # are read in as comments
#
# currently the repository can be configured to support acoustic,
# seismic, and pir processing also the number of records held in
# the repository can be set
#
# To setup a repository for processing on a particular channel the
# follow convention is used
#       Chan Processing NumberRecords Gain
#
```

```
# where Chan is a string - Chan_1,Chan_2,Chan3,Chan_4
#
# Processing is a number
#    0 - No Processing
#    1 - Acoustic
#    2 - Seismic
#    3 - PIR
#
# NumberRecords is a number that tell how many records the repository should hold
#
# Gain is the sensoria number
#    0 - 2
#    1 - 11
#    2 - 101
#    3 - 1001
#
###########################################################################################
Chan_1     1 400 0
Chan_2     1 400 1
Chan_3     1 400 1
Chan_4     1 400 1
High_Level 0 100 1
```

The format of this file is pretty simple, # is used to comment lines out and Chan_1 – Chan_4 are used to configure the input channels. In the example above, channel one is configured for acoustic signal processing, 400 records in the repository and a gain of 2. Channels two through four are setup for acoustic processing, 400 records in the repository and gains of 11.

Polling Example

The polling example code consists of several files, which are located in the jim_reich directory. A description of each file is given below:

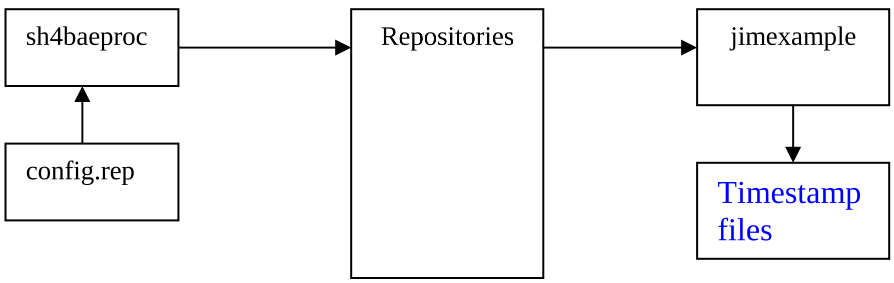
Main.cpp – example code for using the functions in RepPoll.cpp

RepPoll.cpp – contains calls that poll the repository for new time series data and signal processing data. These function calls will retrieve consecutive records from the repositories so that all the records have the same time stamp.

RepPoll.h – header file for RepPoll.cpp

MakeFile – make file for example code. It creates a file called jimexample and puts it in the bin directory.

To run the example code copy the sh4baeproc code to one of the sh4 nodes along with its required files, copy jimexample to the node, and set the config.rep file up so that it does acoustic processing on all four channels. Sh4baeproc can now be started from the command line. When it is started it will read the config.rep file and build the repositories in /tmp, it will also start reading data from the a/d channels and putting data into the repositories. The jimexample program can now be started. When it is started, it will detect that the repositories are already being used and that all it needs to be is just connect. It will then call the InitRepPolling function that will setup static variables used during the polling process. Two other functions are called in this program. The first is GetAcousticTimeSeries and the other is GetAcousticSeriesfft. These functions return true when new data is available and return false when no new data is available. Each time new data is available the time stamp of the received record is written to a file. The GetAcousticSeriesfft function call will return data at one-fourth the rate of the GetAcousticTimeSeries function. This is because four time series records are used to produce one signal processing record. The diagram below is used to illustrate the interaction described above.



To compile a program that uses the repositories include one or all of the following header files: RepUtil.h, RepTypeSafe.h, RepPoll.h and link in Sh4RepLib.a. If RepPoll.h is included, also compile in RepPoll.cpp. To compile the repositories for the pc, comment out all of the sh4 compiler stuff at the top of replib.mk and use make –f replib.mk RepLib. Sh4RepLib.a can be found in sh4lib and replib.mk can be found in /mmapreps.

DataSender

Datasender is a program that we use to log and view data. This program uses subscriptions to get data from the repositories and it is configured with the datasender.cfg file. Typically what we do is to mount a network drive on one of the nodes and then run the sh4baeproc program to put data into the repositories and then run the datasender program and have it open a file on the mounted drive and then log data to it. If people want more documentation on this program they can e-mail me at **brian.corser@tracor.com** or call me at 512-929-4082.

Splitter

Splitter contains code for a program called split. This program will split the files generated by the datasender program into individual files. To run this program first make a directory called splitdata and then type split filename. The split data will be put in the folder splitdata. Again if more documentation is needed let me know.

ProcStruct.h

```
//
// ProcStruct.h
//
//////////////////////////////////////////////////////////////////////
#ifndef __ProcStruct_h__
#define __ProcStruct_h__

#ifndef SAMPLES_PER_SECTION
#define SAMPLES_PER_SECTION 256
#endif

#define SAMPLES_PER_SECTION_ACOUSTIC      SAMPLES_PER_SECTION
#define SAMPLES_PER_SECTION_ACOUSTIC_FFT  2*SAMPLES_PER_SECTION

#define SAMPLES_PER_SECTION_SEISMIC      SAMPLES_PER_SECTION
#define SAMPLES_PER_SECTION_SEISMIC_FFT  SAMPLES_PER_SECTION/2

#define SAMPLES_PER_SECTION_PIR         5

//////////////////////////////////////////////////////////////////////
//
//  Enumeration for the different types of structures that can be
//  stored in a repository
//
//////////////////////////////////////////////////////////////////////
typedef enum
{
        REP_ACOUSTIC_TS,
        REP_ACOUSTIC_FFT,
        REP_SEISMIC_TS,
        REP_SEISMIC_FFT,
        REP_PIR_TS,
        REP_HIGH_LEVEL,
} REP_STRUCT_ID;

//////////////////////////////////////////////////////////////////////
//
//  Structure for holding record header information
//          all the records that BAE stores in the repositories will
//  contain this header as the first element their structure
//
//////////////////////////////////////////////////////////////////////
typedef struct
{
        timeval      ts;      // timestamp for this set of points
        int          gain;    // amplifier gain
        float        rate;    // sample rate
        REP_STRUCT_ID structid; // identifies type of struct in the repository
        int          numpts;  // number of points in this record
} REP_RECORD_HEADER;

//////////////////////////////////////////////////////////////////////
//
```

```
//  Structure for holding Acoustic time series data
//
//////////////////////////////////////////////////////////////////////
typedef struct
{
	REP_RECORD_HEADER hdr;

	float   data[SAMPLES_PER_SECTION_ACOUSTIC];
} AcousticSection;

//////////////////////////////////////////////////////////////////////
//
//  Structure for holding Acoustic psd data
//
//////////////////////////////////////////////////////////////////////
typedef struct
{
	REP_RECORD_HEADER hdr;

	float   data[SAMPLES_PER_SECTION_ACOUSTIC_FFT];
} AcousticSectionfft;

//////////////////////////////////////////////////////////////////////
//
//  Structure for holding Seismic time series data
//
//////////////////////////////////////////////////////////////////////
typedef struct
{
	REP_RECORD_HEADER hdr;

	float   data[SAMPLES_PER_SECTION_SEISMIC];
} SeismicSection;

//////////////////////////////////////////////////////////////////////
//
//  Structure for holding Seismic psd data
//
//////////////////////////////////////////////////////////////////////
typedef struct
{
	REP_RECORD_HEADER hdr;

	float   data[SAMPLES_PER_SECTION_SEISMIC_FFT];
} SeismicSectionfft;

//////////////////////////////////////////////////////////////////////
//
//  Structure for holding PIR time series data
//
//////////////////////////////////////////////////////////////////////
typedef struct
{
	REP_RECORD_HEADER hdr;

	float   data[SAMPLES_PER_SECTION_PIR];
```

```
} PirSection;

////////////////////////////////////////////////////////////////////////
//
//  Enumeration for high level repository structure
//
////////////////////////////////////////////////////////////////////////
typedef enum
{
        REP_ACOUSTIC_SENSOR,
        REP_PIR_SENSOR,
        REP_SEISMIC_SENSOR,
} SENSORID;

typedef enum
{
        BAE_AUSTIN,
        PSU,
        WISCONSIN,
} SOURCEID;

/////////////////////////////////////////////////////////////////////////
//
//  Structure for holding event detection information
//  high level repository
//
/////////////////////////////////////////////////////////////////////////
typedef struct
{
        REP_RECORD_HEADER hdr;

        long     nodeid;     // unique node identifier
        timeval   timestamp;  // timestamp for the detection
        SENSORID sensorid;   // identifies the sensor that made the detection
        int      targetcode; // code book value for the target
        SOURCEID sourceid;   // identifies who generated the detection
        int      direction;  // direction the target is traveling
        float    speed;      // target speed at cpa
        float    range;      // target range at cpa
        float    confidence; // confidence value of the detection
} HighLevel;

#endif
```

RepTypeSafe.h

This header file has type safe called for adding and retrieving data from the repositories.

```
//
// RepTypeSafe.h
//
/////////////////////////////////////////////////////////////////////////////

#ifndef _RepTypeSafe_h_
#define _RepTypeSafe_h_


#include "RepUtil.h"


void RepAddAcousticTs(REP_ID id,AcousticSection *prec);
void RepAddAcousticfft(REP_ID id,AcousticSectionfft *prec);

REP_RESULT RepGetAcousticTs(REP_ID id,int index,AcousticSection *prec);
REP_RESULT RepGetAcousticfft(REP_ID id,int index,AcousticSectionfft *prec);


void RepAddSeismicTs(REP_ID id,SeismicSection *prec);
void RepAddSeismicfft(REP_ID id,SeismicSectionfft *prec);

REP_RESULT RepGetSeismicTs(REP_ID id,int index,SeismicSection *prec);
REP_RESULT RepGetSeismicfft(REP_ID id,int index,SeismicSectionfft *prec);


void	RepAddPirSection(REP_ID id,PirSection *prec);
REP_RESULT RepGetPirSection(REP_ID id,int index,PirSection *prec);


void	RepAddHighLevelEvent(REP_ID id,HighLevel *prec);
REP_RESULT RepGetHighLevelEvent(REP_ID id,int index,HighLevel *prec);


#endif
```

APPENDIX C: BAE Low Level Signal Processing

Overview

The purpose of this document is to provide documentation for the low-level signal processing code along with the changes that have been made to the repository code. A detailed presentation of the signal processing chain can be found in BAEProcSpecVersion1.3.ppt. This document will mainly focus on setup, compiling and running the supplied code.

Signal Processing Code

The aim of the low-level signal processing code is to provide appropriate processing for the current sensor suite. Specifically, we are providing acoustic, PIR and seismic processing. The default configuration is to have acoustic processing on channel 1, PIR processing on channel 2 and seismic processing on channel 3. It is not currently possible to change this default setting, but this capability will probably be added in the future. In order to make the result of the low level processing available to other groups, they are being written out to the repository. Documentation has already been provided that covers the basic workings of the repositories and so it will not be covered again here. The next section will describe the changes that have been made to the repositories and describe the structures that we are storing in them.

The signal processing code currently sets the sample rate to 1 and the gain for each channel to 1. This will be changed in the future to use BBN’s node initialization code. Also, in order to run, the following files will need to be present:

HPFfiltcoefs.dat
Hanning1024.dat
Hanning256.dat
dsfiltcefs64.dat.

Repositories

Several changes have been made to the existing repository code and are listed below.

1. Fixed a bug that sometimes prevented a client from properly connecting to the repository.
2. Added a configuration file so that the type and number of records held in a repository can be configured.
3. Defined specific structures for each type of signal processing.
4. Added type safe interface to add and retrieve data.
5. All repositories will now be put in the /tmp directory instead of the current directory.

config.rep is used to setup and configure the repositories and will be located in the /tmp directory. The idea behind the config file is that each channel has a time series and a signal processing repository which can be configured to hold the results of either acoustic, PIR or seismic signal processing. If config.rep is not present when a client connects a default configuration for the repository will be created with acoustic processing on channel 1, seismic processing on channel 2 and PIR processing on channel 3. A default config.rep file will also be created.

The structures for each type of processing are in ProcStruct.h.

APPENDIX D: CONFERENCE PAPERS

Collaborative signal processing for a cluster of inexpensive distributed sensor nodes

Steven D. Beck, Joseph Reynolds

BAE SYSTEMS Integrated Defense Solutions Inc.
6500 Tracor Lane MS 1-8
Austin, TX 78725
512-926-2800
steve.beck@baesystems.com
joe.reynolds@baesystems.com

The tactical and urban warfare community has a significant need for up-close sensing capability along with increased sensing coverage. The recent availability of small, inexpensive, light weight, and low power sensing nodes is providing the means to cover an area with a distributed network of remote intelligent sensors. Utilizing local node information combined with information from neighboring nodes is a significant challenge. An even bigger challenge is making use of meta-knowledge and tasking a cluster of nodes to cooperate on a task specified at a very high level. This paper provides a survey of collaborative signal processing considerations and techniques that can be used with a distributed network of smart sensor nodes to improve target detection, identification, localization, and tracking. Factors for system design are discussed, along with an example application.

Keywords: Collaborative Signal Processing, Distributed Sensors, Optimal Detection, Fusion, Meta-Knowledge

1. INTRODUCTION

This paper discusses aspects of sensing for tactical military objectives, in contrast with military strategic sensing, environmental monitoring, or other examples. System level objectives include detection and tracking of motorized or dismounted forces, threat indicators and warnings, and timely updates to the tactical picture. The objectives require measuring temporal relationships among individual detection events on a short time scale, call for decomposition and recombination of event sequences, require establishment of geographical relationships, and systems must work in the presence of countermeasures. Obviously, these objectives shape the processing and the node architecture.

Consider the objectives in complex or urban situations. Tactical sensing must detect and characterize an opposing force without contact [find them before they find you and start an ambush]. Commanders need to discover the opposition's axis of advance, their extent and their front. They must detect concealed forces and establish a positive identification before counter fire. In cases of lower intensity combat or Operations Other than War, the commanders must provide both area and point surveillance, watching markets or bridges or key intersections, without undo risk to soldiers from snipers and bombers. Threats may mix with civilian non-combatant forces or mobs may be the threat. Sensing must be performed where there are hills, canyons, forests, buildings, sewers, and tunnels.

The time scale governs many aspects of tactical sensing. In high intensity combat situations forces move within the (re)targeting time for aircraft or artillery, on the order of 10-30 minutes. Sentry and perimeter security for a command post, possibly a parked vehicle, has a scale of minutes. If monitoring crowds or threat surges the times are hours. Only for monitoring of compounds, snipers, or infiltration routes do times extend to days. In each of these cases targets are only active for tens of seconds to a few minutes. For example, in seconds a soldier may move to the window and fire. He can drive from the compound gate at 30mph and be half a mile away in a minute or around a corner in 12 seconds.

Incorporating knowledge about the world in which the sensors operate is also enormously important to their success. This includes knowledge of the immediate tactical situation, what forces are where, their state and intent; knowledge of the environment, especially the terrain and cultural artifacts such as roads; knowledge of what other neighboring sensors have detected; and self knowledge including location, battery state, and data link effectiveness. We call this Meta-Knowledge and using it is one key to successful sensing.

2. SENSING ARCHITECTURES

2.1 In-situ sensing vs. remote sensing

In order to improve detection, target tracking, and tactical inference, there are two basic sensing architectures. The traditional architecture is what we will call 'remote' sensing in that the sensor system is separated from the contact space. This is the organization of air search radars, shipboard sonars, observers with binoculars, and "spy" satellites. The architecture we will discuss in this paper we call 'in-situ' sensing in which the sensors and the contacts are co-located in the same space. We discuss situations with sensors on street corners, sensors along roads or pathways, and sensors in/on buildings. Further, we emphasize fields of sensors and linked / cooperative sensors. Common examples of in-situ sensors are home security alarms, mine fields, and sonar buoy fields.

Traditional methods of remote sensing and collaborative processing include array processing (beam forming, etc.), multi-sensor information fusion, and data association and tracking. Implementation of the algorithms requires systems with high communications bandwidth and few limits on processing power or memory. These systems often have a single point of failure, e.g. if one element in a beam forming line array goes out, the beam patterns will be significantly degraded. Large calibrated systems require hand emplacement and longer range radios that make these systems more vulnerable to attack. On the other hand, a dense spatial array of networked sensing nodes can be designed with local detection processing and distributed decision making. When designed properly, the In-Situ approach provides graceful performance degradation as nodes become unusable. Table 1 below provides a basic comparison of these two systems.

System Description	In-Situ Sensing	Remote Sensing
Single Node Capability	Low	High
Fault Tolerant	Yes	No
Performance Degradation	Gradual	Sudden
Bandwidth Requirements	Low	High
Power Requirements	Low	High
Cost	Low	High
Size	Small	Medium to Large
Emplacement	Random	Precise
Range	Local	Local or remote

Table 1 - Comparing in situ with remote sensing

2.2 Sensor node characteristics

If we are discussing a field of sensor nodes, what is each node like? Are all nodes the same? What is important about the nodes? In this paper we assume that each node has these eleven components:

1) Sensing Transducers - these are geophones, microphones, imagers, and similar items that are chosen to support the problem at hand and the environment in which the node is deployed, including their interfaces and data acquisition;

2) Local Processing - a computing capability suited to required performance and to allowed power budget;

3) Local Storage - memory for programs, parameters, signals, intermediate processing products, environmental description, tactical situation meta-knowledge, data bases, and possibly shared items for remote access;

4) Node - Node Communication - two way data links between nodes in an area for sharing of data and for coordination of processing;

5) Node - User Communication - two way data links between nodes and user terminals for sending results and for receiving queries, processing parameters, tactical meta-knowledge, and configuration commands;

6) Node Location - knowledge of transducer and transmitter location, locally determined or externally set, with accuracy to support tactical goals;

7) Time / Clock - time of day to precision required for tactical objectives and clock pulses for initiating processing and synchronizing events;

8) User System - the external data universe which interfaces with the node;

9) Security - tamper security, authentication, jam and detection resistance;

10) Software - programs structured to support many time constraints, signal processing, communication, decision sequencing, contact movement intervals;

11) Power Management - on-node sensors to measure and control batteries to maximize mission performance.

Each of these characteristics will have different design parameters depending on how the nodes are to be used and how the field is configured. This paper will not attempt to specify design settings, but will discuss factors effecting the choice of settings.

2.3 Sensor field configurations

Sensor fields can be configured three ways with different architecture impact; unorganized, organized for query, and organized for computing. Each configuration constrains the component characteristics of individual nodes, constrains node-node communication characteristics, and even constrains the precision of node location.

2.3.1 Unorganized

Unorganized sensors depend on the innate local node capability, possibly with simple sharing of like data among adjacent nodes. Typically nodes are uniform in capability and operation. Processing is node centered with each node generating detection and classification events. Node-node communication is usually limited to simple exchanges, alerting neighbors to wake them from a low power "sleep" state, and sharing signal processing results among nodes to sequentially improve detection through longer exposure or through more independent 'looks.' The user fuses individual detections into high level intelligence after observations are retrieved from the field. Nodes are isolated for the complexity of tactical situations and not affected by the number of users or the number of retrievals.

This is the simplest configuration and makes the least demand on individual node architecture and performance requirements. Difficult computing is performed in the user environment where there are more resources. Node processors can be safely sized to the signal processing algorithms. Memory can be sized to meet real time processing, which is limited by the physics of the detectors. Communication need be only sufficient to send detection events to users on demand. Node-to-node exchanges are not more complex than the user query. Responses can take tens of seconds. The simplicity of this configuration permits use of very simple radios and software.

2.3.2 Organized for query

A configuration organized for query explicitly selects nodes from a field to meet the needs of identifying a predetermined condition. Meta-knowledge of goals, the field environment, and contact characteristics is used to select the nodes. Nodes may become "distinguished" and perform unique functions. Processing on nodes is selected to match the situation, changing node to node and query to query. Tactical knowledge is sent to the nodes as queries are posed. The field of selected nodes fuses results to derive intelligence as specified by each query. Nodes exchange complex reports; signal processing and decision processing items, including descriptive data for current situations, node status as to software configuration, power status, and communication neighborhood listing. Because of local reconfiguration nodes are not isolated from the complexity of the user's situation, and must be capable of simultaneously processing several queries at any time for different users with different intents.

This configuration makes significant demands on node software complexity and communications. Node processors must be sized for both signal processing and for higher level information processing. It is likely that floating point arithmetic is required. Multi tasking of software implies memory and state resource mapping hardware in the processor. Memory of each node must be sized for anticipated levels of simultaneous queries, each query with unique program and data storage. On-node storage may be served by a local database shared among tasks. Communication among nodes is much more complex than communication with users; a local network is likely required. Responses must meet time horizons of decision algorithms, including any node-node network latency.

2.3.3 Organized for computing

A configuration organized for computing attempts to form a distributed parallel computer by associating selected individual nodes and connecting them via local communication. Several such associations may be in force simultaneously, each sharing some common task such as signal processing dedicated to a temporary objective. Each chosen association matches node capabilities to algorithm requirements. One example is selection of nodes to form a detector array suited to beam forming, some nodes supplying sensor elements, some primarily processing, and with additional nodes acting as memory.

This configuration makes even higher demands on node software complexity, on communications, and on memory design. All the problems encountered building traditional parallel computers are now encountered in a tactical environment. All the problems encountered building sensor arrays must be solved for imprecisely placed nodes in uncertain local environments. In particular, problems encountered in accurately placing or knowing the location of the node/sensor to within fractions of acoustic or seismic wavelengths must be solved to perform beam forming - not knowing this significantly degrades performance and requires very difficult adaptive algorithms to adjust parameters and search for the right settings. Even if beamforming is not attempted, node - node communications must meet stringent requirements for data bandwidth and low latency. This is one of the classic problems in parallel computers and is very, very difficult in stealthy, low power, packet radio. Similarly node memory design must accommodate remote access by other nodes which share some data item, increasing the required memory bandwidth many fold. Some data item, possibly a semaphore used to synchronize a protected section of shared code or data, will be accessed by many nodes in the array. Access to the item will limit overall performance. To reduce such access overload one strategy has been to replicate data to several (many) nodes. Replication reduces the hardware performance constraint at the cost of synchronization; the copies must be kept consistent within algorithm time budgets. Synchronization imposes a further communications cost and memory size cost for the extra copies. The particular solution depends on details of the parallel algorithm. If the shared item is small, like a semaphore, size is not an issue, but synchronizing copies of a semaphore is very costly due to the update frequency. For a general purpose system both communications and node memory design must be of the highest capability.

2.4 Choosing sensing modes

Obviously sensing transducers must be selected which match the phenomena exhibited by contacts of interest. Most things make noise and many threats are loud so audio is a useful sensing mode for many contacts. Many threats travel on the ground and seismic is a useful mode to detect them. Once operational indicators and warnings are specified, sensing modes can be readily chosen. Table 2 shows characteristics we have found useful.

Since contacts often are detectable in several modes, some design trades are required to select modes to be used against them - vehicles may require a trade of seismic, audio, and passive infrared. Decisions are required as to using active or only passive sensors. Deployment modes, how nodes are to be emplaced, may constrain sensor choices. Consider an example where vehicles are to be detected. Vehicles exhibit in seismic, acoustic, IR, and chemical modes. Since geophones require firm contact with the surface (a spike into the ground, buried, etc.) air deployment may lead to elaborate designs to ensure insertion, especially in areas where pavement is prevalent. These considerations may lead to choosing the simpler audio sensing which is not similarly constrained. Air deployment may remove IR from consideration as the node may land behind an obscuring object. This example points to the need for system design to support a sensor’s expected area of regard. A mode which may be limited by a degraded range of detection should be avoided. However, some sensors have a very large area of regard, for example RF intercept, and this complicates their use since nodes may have overlapping detection zones - complicating localization algorithms and generating redundant detection events.

Modality	Cost	Area Of Regard	Indicators	Constraints
Audio (Single & Multi Chan)	$3/Chan	Omni, to 100m	Persons, crowds, vehicles, velocity, explosions, sentry	Few, Acoustic Clutter forces extensive processing, SVP near surface constrains range, HF absorption, Wind noise
IR/Image (10F/S VGA)	$25	Line of Sight to 100m	Movement, characterization of object by shape, sentry	Obstructions require manual emplacement
Geophone	$20/Single Axis/Chan	Omni to 200m	Movement on Surface, floors or roads, sentry	Manual emplacement, must have firm contact, Disrupted by volume disturbance (ditch) , stairs, construction joints
Motion (PIR & UltraSound)	$7	1m to 20m	Excellent for Movement, Volume Change, sentry	Obstructions require manual emplacement
Laser Detection	$50	LOS to 2Km	Threat Technology, Red/Gray classification	Low power LOS, High Power uses multipath
Radar Detection	$50	LOS to 2Km	Threat Technology, Red/Gray classification	Obstruction

RF (Detect by Band)	$35	Varies, requires control	Threat Technology, Red/Gray classification, discriminator, timing and event association.	Range & Direction Ambiguity
Chem/Bio	$300	Downwind	Aerosol Threat	Few
GPS	$20		Node component, Location to cm Clock to nS	View of Sky when emplaced, co-recording at surveyed site for precision

Table 2 - Common sensing modes & their characteristics

3. PROCESSING & ALGORITHM CONSIDERATIONS

3.1 Detection and classification processing

Typical military and surveillance audio applications include speech detection and localization, gunshot detection and localization, ground vehicle detection and identification, and aircraft detection and identification. All of these applications can produce a signal event containing more energy than that present in the normal background. In these cases, it is possible to "detect" the event simply by determining when the signal energy crosses a threshold. This is the classical optimum detection of signals in noise problem. However, this simple detection scheme presents several difficulties in real-world applications.

The first problem with simple detection is that there is not a single constant definition of normal background noise, especially in an outdoor environment. The noise environment is not stationary in either time or frequency content, and varies with respect to season, time of day, temperature, humidity, and many other environmental factors. Instead of attempting to calculate an adaptive threshold, a more productive approach is to adaptively normalize the input data. Signal normalization can be performed across both time and frequency, which if done properly, can provide a constant false alarm rate (CFAR) of detection with a single uniform threshold level. This type of normalization is essential for detecting weak continuous signals and non-traditional signals such as transients.

CFAR normalization requires knowledge of the probability density function of the noise at each frequency and time instance. The windowed short-time Fourier transform (STFT) is used to produce a linear scale tiling of the joint time and frequency distribution of input signal energy:

$$Y(t,f) = \sum_{\tau=-N/2}^{N/2} w(\tau - t)\, y(t) e^{\frac{-j2\pi ft}{N}}$$

where w is the window, y is the input signal, and N is the input period. In order to use a constant and uniform threshold, the noise power should be normalized to one, and the mean should be zero. The normalized noise background is then:

$$Z_n(f,t) = \frac{Y_n(f,t) - m_n(f,t)}{\sigma_n^2(f,t)}$$

where

$Z_n(f,t)$ is the normalized noise power spectral value at time t and frequency f

$Y_n(f,t)$ is the input noise power spectral value at time t and frequency f

$m_n(f,t)$ is the mean of the noise power spectrum at time t and frequency f

$\sigma_n^2(f,t)$ is the variance of the noise power spectrum at time t and frequency f

Because the noise is nonstationary and superimposed signals bias any statistical calculation of the background noise, the normalization values must be estimated using some relatively sophisticated robust statistical approaches. Several approaches that were pioneered for narrowband detection in underwater acoustics includes the two-pass split mean normalizer, and the three pass peak sheared normalizer. These techniques first estimate a detection statistic, perform a nonlinear operation on the data (e.g. shear any peak over the threshold,) and then re-estimate the detection statistic.

BAE SYSTEMS has successfully improved on these normalization techniques by using adaptive multi-resolution gamma filters 1-2 for estimating the detection statistics, and simple trackers for indicating transients verses the onset of longer period background processes. The gamma filter is a simple feedforward digital filter with gamma functions as the tap coefficients:

$$g(n) = \mu(1-\mu)^n$$

where g(n) is the coefficient at tap n and is the smoothing coefficient. This simple filter provides an efficient means of trading off temporal resolution for system memory.

The primary method of determining the operating point of a detection system is by plotting the Detection Error Tradeoff (DET) curve from an empirical set of log likelihood ratio measurements. The DET curve plots the probability of a missed detection (Pm) in percent on the Y-axis, and the probability of false alarm (Pfa) in percent on the X-axis. A system with perfect detection and no false alarms would have an operating point at the origin. Instead, the operating point is chosen based on the equal error rate (Pm=Pfa), or a fixed cost ratio between Pm and Pfa. The DET curve for acoustic detection of a heavy wheeled vehicle from a 1024 Hz sampling system is shown in Figure 3.

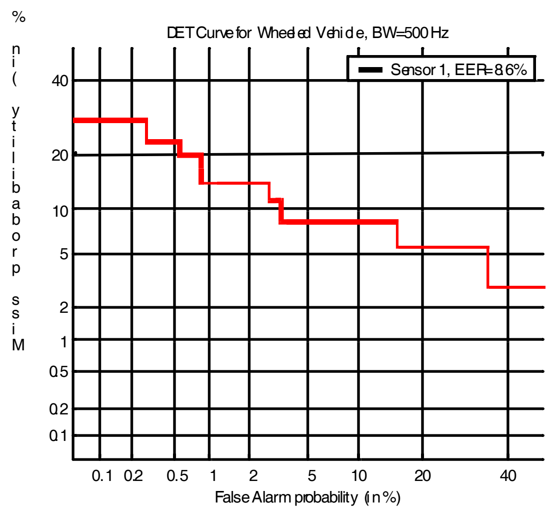


Figure 3 – Example DET curve for a heavy wheeled vehicle

3.2 Signal signatures

Identification of contact signals is a continuing problem for sensor systems. The effort expended in this area over the life of a military sensor exceeds the effort to develop and deploy sensor equipment. Success depends on understanding the physics of signal generation and transfer through the sensing medium. As discussed in Section 2.4, not all sensor designs are equal. The expected capabilities of sensors can be used to enhance classification capabilities and algorithms.

As an example, BAE SYSTEMS has developed signal processing, detection, and classification software for the DARPA/ITO SenseIT Program. The sensor nodes are described later, but have two possible programmable configurations; single channel sensing at 1024Hz and four channel sensing at 256Hz/channel. Sensors include geophones, microphones, and passive infrared (PIR) motion detectors. Experience with reference signal databases and with specially collected data (Figure 4) lead to the following expectations. In one channel processing mode, the sampling rate for that channel is 1024 Hz. Only channel 1 or 2 are eligible for single channel processing. In this case, both detection and classification will be performed and output to the detection and High Level Repositories for multi-node decision making. In four channel processing mode, the sampling rate for each channel is 256 Hz. In this case, the analog bandwidth after the anti-alias filter will be 80 Hz. There is little acoustic energy in the 0-80 Hz band for any of the desired targets in the microphone channel. We will process a detection for the microphone channel in this case, but we expect that wind noise will dominate. Therefore, the primary detection and classification channels are listed below.

Fs	Sensor	Detection	Classification
1024	Microphone/Acoustic	Yes	Yes
1024	Geophone/Seismic	Yes	Yes
256	Microphone/Acoustic	Yes/No	No
256	Geophone/Seismic	Yes	Yes
256	PIR Motion	Yes	No

For the single channel acoustic and seismic processing, the high level detection and classification outputs will rely on finding the closest point of approach (CPA) from the data record. For the multi-sensor configuration, the CPA can be estimated from the data record. However, a more reliable CPA estimate will result from using the directional PIR detection. We attempt to use the PIR detection as the estimate for CPA time when writing to the High Level Repository.

Some of the expected targets will have detections from a number of different sensing modalities. The number and type of sensing modality may aid in the classification process. A matrix of preliminary targets and expected modality detections is given below.

Class / Sensor	Mic 1024	Mic 256	Seismic	PIR
Wind	Yes	Yes	No	No
Clutter	Yes	No/Yes	No	No
Voice	Yes	No	No	No
Person	No	No	No	Yes
Vehicles	No/Yes	Yes	Yes	Yes

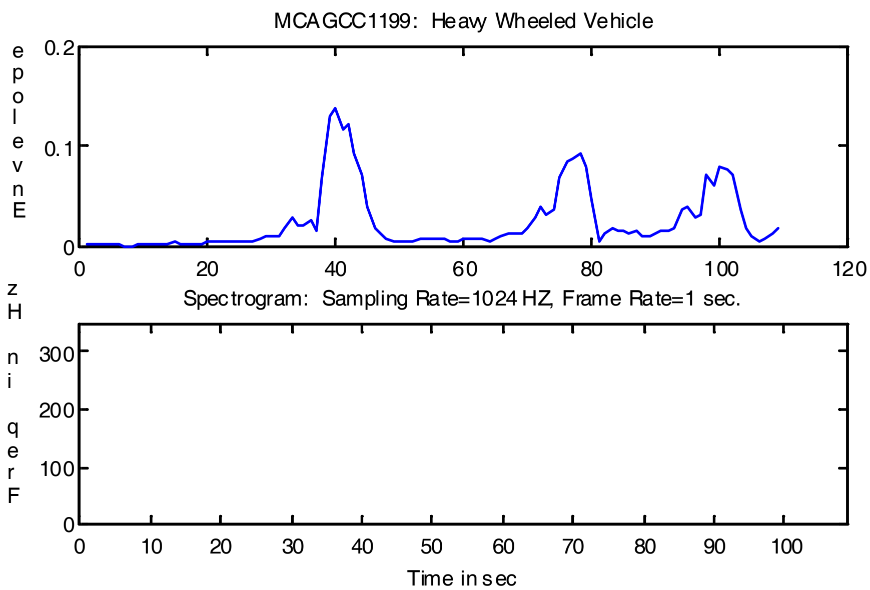


Figure 4 - Signature for a heavy wheeled vehicle

3.3 Local and distributed decision making

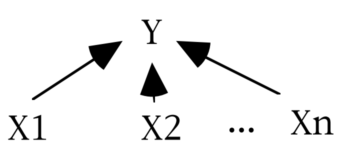
Supra-Bayesian information fusion combines probabilistic information from multiple sources into a single improved decision, and analyzes the correlation among the inputs.[3-5] A graphical network structure provides a description of the causal relationships among the input and output variables. Figure 5 shows an example of a possible probabilistic network for acoustic target detection. Each node in the graph represents a random variable (input, hidden, or output), and connections between nodes describe the causal relationships among variables by their representative conditional probabilities. The conditional probabilities, or likelihood functions, can be calculated using cross tabulation for discrete variables (the *Bayesian network*) and gradient descent for the continuous case (the *adaptive probabilistic network*). When measurements become available for some of the nodes, the posterior probabilities for the other nodes, including the output nodes, can be efficiently computed. The precise, local, and probabilistic interpretation of these networks provides a sound mathematical basis for making inferences under uncertainty, and allows partial or whole construction of the network topology by experts.

Local node detection and classification can be incrementally improved by the use of *probabilistic (Bayesian) networks*, which have been used successfully in many signal classification and information fusion applications. Probabilistic networks are directed graphs that give an explicit representation of the joint probability distribution characterizing a problem domain. These networks implement Bayes' rule for the joint distribution expressed using conditionally independent random variables:

$$P(x_1, \ldots, x_N) = \prod_{i=1}^{N} P\big(x_1 \big| Parents(x_i)\big)$$

An example of a two variable, a three variable, and a poly-tree structure of causally related random variables demonstrating conditional independence is given below:

Causal Relationship	**Joint Distribution**
$X \rightarrow Y$	$P(XY) = P(Y\|X)P(X)$
$X \rightarrow Y \rightarrow Z$	$P(XYZ) = P(Z\|Y)P(Y\|X)P(X)$



$$P(XY) = P\big(Y \big| X_1, \ldots, X_N\big) \prod_{i=1}^{N} P(X_i)$$

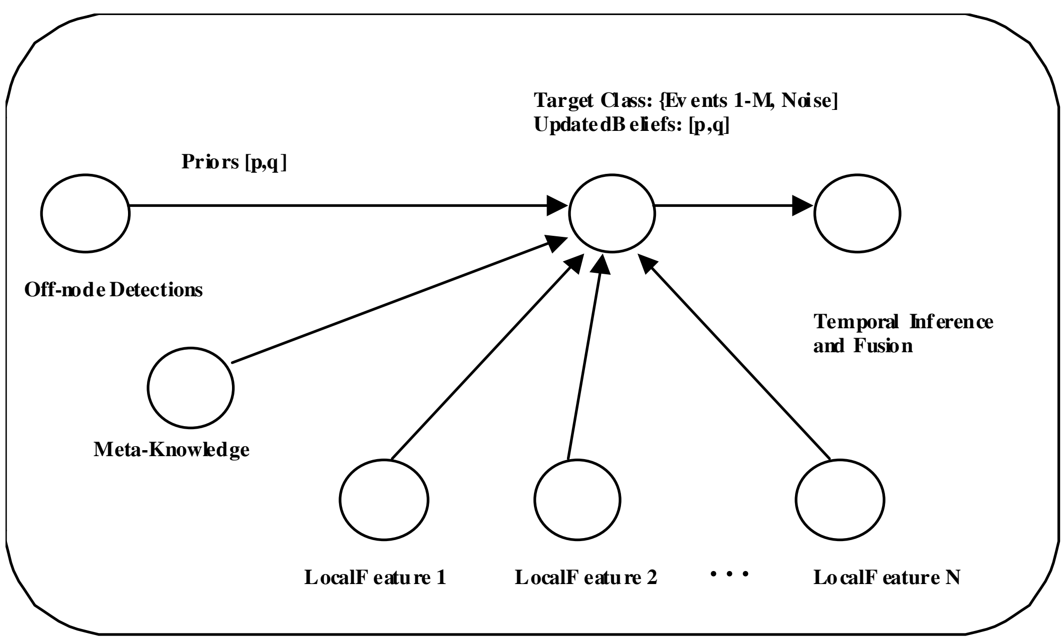


Figure 5. Example Probabilistic Bayesian Network applied to distributed detection

3.4 Architecture considerations of signal and information processing

We assume that the reader is familiar with design processes and architecture considerations of signal processing, sizing arithmetic, data acquisition and analog components; these are mature techniques. However several items deserve discussion for fields of sensors, especially those organized for queries.

The first special design item is provision for on-node repositories. These are organized storage of processing parameters, intermediate results, detection/classification/localization events, and meta-knowledge. We found that building on-node databases with qualified updates and queries is worth the resources. Repositories have multiple index fields of formatted data to accommodate regular retrievals, and binary large objects (BLOBS) for mostly unstructured data such as time series. While simpler table or file storage might use less resources, we found that the uncertain nature of tactical sensing made the functionality of databases worth the extra programming and processing. The more formal definition of repository entries made data interchanges much more efficient and saved communications. We built Time Series Repositories for signal processing algorithms and this permits easy sharing among neighboring nodes of data for concurrently held targets. We built Information Processing Repositories for detection and classification algorithms. They proved useful when fusing contacts held on several sensing channels. We built High Level Repositories for extending localization across nodes and for responding to queries. We built Node Status Repositories for items describing individual nodes. Finally, we built Meta-Knowledge Repositories for a wide range of tactical and operational items.

Another important special design item is time processing. Nodes require time coherence to allow contact fusion, whether on-node or across nodes. We settled on UTM as defined by our GPS receivers. Care must be exercised when processing time under software in PC development frameworks. The presentation format may not match the clock capabilities, with millisecond or microsecond display precision, but a clock period of 16ms or 20ms. The developer should investigate the software framework in the target hardware environment as the framework may adapt to its operational capability.

3.5 Meta-Knowledge and tactical situation awareness

Meta-Knowledge is any collection of operational knowledge or intelligence parameters which describes parts of a sensor network, its environment or condition, and which are not embedded in a fixed way in the implementation, but supplied by external sources such as operators and legacy data systems; including knowledge discovered by the network during operation. This definition contrasts with parameters embedded in algorithms. Some meta-knowledge is configuration data (for instance sensor type to channel assignment) supplied by an operator. Other operational parameters may be obtained by the node without operator support; like battery state or time or node GPS location. The most useful meta-knowledge is about tactical and environmental items which affect decision and information processing algorithms. Items from the physical environment such as signal transmission conditions (wind speed which affects both expected acoustic noise characteristics and dust levels) and from the geographical environment (such as the presence of roads or barriers) are particularly useful in qualifying detections, classifications, and localization. They are useful in the layers of processing beginning after initial signal processing through initial Bayesian decisions. Meta-knowledge items from doctrine of the opposition and of own forces are useful in multi node inferences and inferences over an extended time or history. Unorganized sensor fields, those depending on a user terminal to integrate multiple nodes, often incorporate meta-knowledge in informal ways through the operator's reaction to his knowledge. User query terminals for an organized field also incorporate meta-knowledge informally. The content and sequence of queries will be made in context of a tactical picture held by the user. Incorporation of meta-knowledge into the processing of autonomous sensor fields is a work in progress. Table 6 shows some items to consider.

Class	Instance	Values
Tactical Situation	Force Structure, listing forces and composition	Equipment in the Field, qualified by source, location, and time.
Contact Naming or Detection Code Book	Common Target Type Identifier	Code Book Value for efficient exchanges
Operational Settings	Detection & Classification Signatures	Signal Processing Parameters
	Decision Criteria	Thresholds or coefficients
	Area of Regard	Radio or Sensor expected range or coverage
Location	Named Place	Geographic Reference
	Operational Place	User Terminal Location, Force Hq Location
Neighborhood	Administrative Region, for instance 1MAR Sector, or Omaha Beach Red	Geographic Reference(s) - usually a list of coordinates but occasionally other formats

	Collaboration Region, for instance "Along the Road"	Geographic Reference(s)
Collaboration Assignments & Settings	Scripts	Processing Orders, Node Order, Neighborhoods, Operational Settings
	Neighborhoods of Interest	Names
Sensing Environment	Terrain Characteristics	Terrain & Cover, Elevation, Barrier(s), Land/Water
Social Artifacts	Roads, Dams, Buildings, Towers	Code Book Value, Descriptions
Separation Criteria	Geographic Barriers	River, Chasm, Forest, Mountain, Building
	Target Characteristics	Requires Road, Moves through forest (or not), Flies over
Confirmation Criteria	Signal or Processing Changes	Slowing for Corner, Climbing a Hill; matching evidence encoded as representative of the operational area

Table 6 - Meta-Knowledge examples

4. EXAMPLES

4.1 Use of Meta-Knowledge in a query processing example scenario

A Sensor field organized for query processing is defined in Section 2.3.2. By explicitly selecting nodes to meet the needs of a query, this organization provides more capability than retrieving from unstructured processing. The query can supply meta-knowledge, such as geography, node relationships, timing and alerting, which improves information and decision processing

Collaboration begins when the query selects nodes of interest and specifies an expected detection and a level of processing. The selection may be geographic , such as nodes "along a road" or "North of a line" or "In 1Reg Sector." Some nodes may be distinguished, such as the start or end of a string, or those on the boundary of a region. The query user terminal has the meta-knowledge required to associate nodes with a query condition, knowing for instance where the road lies and the distance of interest either side the road. The query can have meta-knowledge such as the expected travel direction, or very specific meta-knowledge such as the distinguishing elements of the signature of a particular vehicle, not just a class. The query has meta-knowledge of the significance or criticality of a level or type of processing; being interested in confirming a vehicle ID but having much less interest in its speed. Use of meta-knowledge by a node may restrict processing and thus conserves battery; or expand portions of an algorithm or parts of a chain or number of iterations or allowed error on an estimate. Meta-knowledge will influence, possibly explicitly, which nodes share data, which nodes process, and which report to the user terminal.

Consider a query directed to a sensor field as shown in Figure 7: "Respond with classification and speed if a Command LAV is detected on the road traveling NE toward the intersection". Each sensor can detect and with some reliability classify the vehicle, and can estimate basic limited motion from Doppler. The query must order the nodes such that 'a' is the "initial" node; detection at 'c' or 'd' would indicate the wrong direction of travel. The query must cause the nodes to take the extra step of estimating Time of CPA. Further the nodes must share T_{CPA} such that road speed may be estimated. The nodes must be supplied the extra signature information which distinguishes a 'Command LAV' and they must use it in successive observations to confirm with some certainty the classification. The nodes 'b' 'c' and 'd' must exchange estimates of position and speed such that a diversion to '5' doesn't cause a response. This processing proceeds much as in unorganized collaboration except for the processing and extra algorithms associated with the meta knowledge - direction of travel depends on node ordering, speed depends on successive T_{CPA} and the road length between nodes vs geometric distance, detection of non-diversion (position '5') depends on estimated T_{CPA} being satisfied at 'd' for position '6' and confident classification depends on many observations using extra signature values.

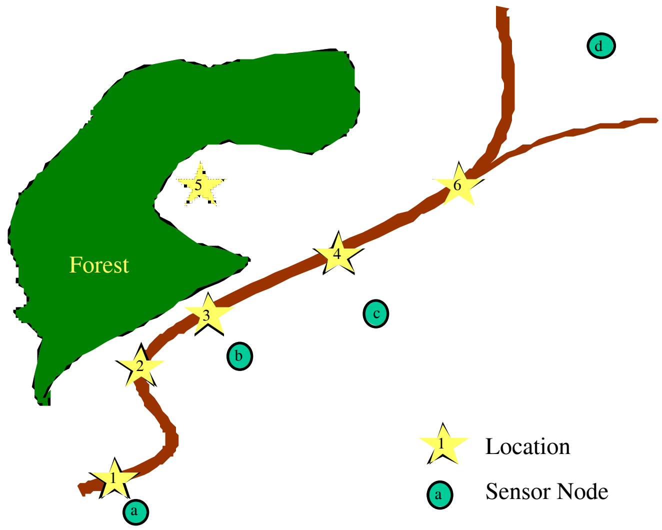


Figure 7 - Example scenario laydown

4.2 Example Systems
SenseIT/WINS nodes are produced by Sensoria Inc and used in the DARPA/ITO SenseIT program first year effort (Figure 8). New generation nodes are proposed for the second year. Each node has GPS for location and clock, and a frequency hopping packet radio for node to node networking. Some nodes are distinguished by a high power radio for long distance gateway service. Each node accepts four transducer channels, typically seismic, audio, PIR, and imager. Operation is either single channel at 1K samples/sec or four channels each at 256 samples/sec. Nodes have two on board processors. A very low power preprocessor performs continuous service for data collection and data link control. Data is linked by serial port to a main processor with program and file storage. The main processor is based on a MIPS R3000 which supplies very good signal processing performance. WINS nodes are designed for extended life once emplaced. The preprocessor runs under proprietary real time control. The main processor runs under Microsoft WinCE (2.x) SenseIT has developed a networked sensor field organized for user query under guidelines described earlier in Section 2.3.2. Software has been developed by the SenseIT program participants but a description of that activity is out of the scope of this paper.

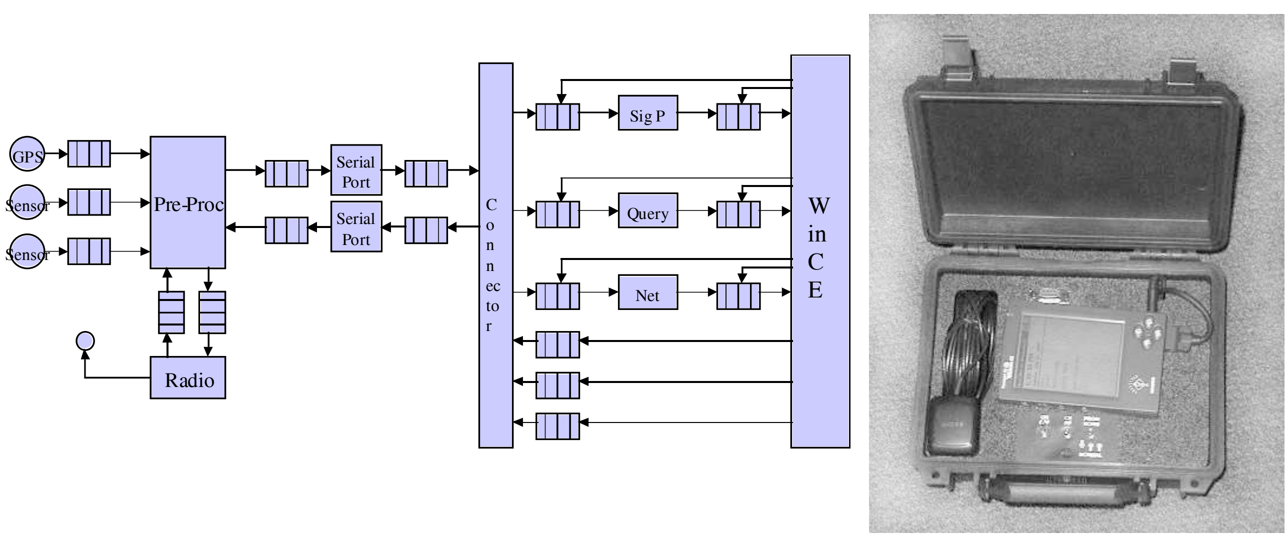


Figure 8- DARPA/ITO SenseIT Program WINS Node by Sensoria Inc.

DIMTASS nodes are produced by BAE SYSTEMS (Figure 9). Each node has GPS for location and clock, and a Manchester coded data link to a user terminal which can control as many as 32 nodes. Each node has audio and video/IR sensors. The on-board processor performs signal processing, detection, classification, localization, and data management with proprietary real time software. Hardware controls data acquisition and radio functions. DIMTASS are very inexpensive miniature nodes designed for short duration tactical applications such as Urban Scout missions, immediate site surveillance, and traffic lane monitoring. They are designed to provide positive identification of exposed threats and to detect and characterize concealed threats. For surveillance the nodes are hand emplaced. For scout and other applications the nodes may be grenade launched or air deployed from UAV or other aircraft self protection dispensers. DIMTASS is an unorganized sensor field as described in Section 2.3.1.

Figure 9 - BAE SYSTEMS DIMTASS miniature disposable nodes & user terminal

5. CONCLUSION

In this paper we have indicated that smart sensor fields are feasible. We have presented design factors for the field's organization, for nodes, and for algorithms, most of which are well understood. We have indicated which design trades we found very important. We have presented processing algorithms that are experimental in this application, but have been useful in similar problems under different conditions. We have described local repositories, on-node databases of processing products, which we found necessary for algorithms which incrementally improve detection, classification and localization, whether node to node across a field of sensors or over time on a single node. Repositories are also necessary when organizing the sensor field to directly respond to queries. Finally, we have discussed incorporating Meta-Knowledge in algorithms to improve detection and classification performance.

6. ACKNOWLEDGEMENT

Work mentioned in this paper has been partially supported by DARPA Information Technology Office, Dr. Sri Kumar, under the SenseIT Program; Contract F30602-99-C-0159.

7. REFERENCES:

1. Principe J.C., de Vries, B., de Oliveira P.G., "The Gamma Filter-A New Class of Adaptive IIR Filters with Restricted Feedback," IEEE Trans on Signal Processing, Vol. 41, No. 2, Feb 1993.
2. S. Beck, L. Deuser, and J. Ghosh, "Robust Classification Techniques for Acoustic Signal Analysis", *Proceedings of the IEEE Signal Processing Workshop on Statistical Signal and Array Processing* , 7-9 October 1992.
3. Jacobs, R.A., "Methods for Combining Experts' Probability Assessments," Neural Computation 7, 1995.
4. S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach. Englewood Cliffs, NJ, Prentice Hall, 1994.
5. J. Ghosh, S. Beck, and C. Chu, "Evidence Combination Techniques for Robust Classification of Short-Duration Oceanic Signals", Proceedings *S.P.I.E. Conference on Adaptive Learning Systems*, Vol 1706, April 1992.